openSUSE Leap 15.7

仮想化ガイド

仮想化ガイド
openSUSE Leap 15.7

このガイドでは一般的な仮想化技術の説明と、仮想化に対する 統合インターフェイスである libvirt の紹介、そして各種のハイパーバイザに 対する詳細な説明を行っています。

発行日: 2026/11/05

SUSE LLC
1800 South Novell Place
Provo, UT 84606
USA
https://documentation.suse.com

Copyright © 2006– 2026 SUSE LLC and contributors. All rights reserved.

訳: SUSE LLC および貢献者が全権利を留保しています。

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled 「GNU Free Documentation License」.

訳: この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書バージョン 1.2 または (希望すれば) 1.3 が定める条件の下で複製、頒布、あるいは改変することを許可します。ただし、この著作権とライセンス表記については変更不可部分とします。この利用許諾契約書の複製物は、「GNU フリー文書利用許諾契約書」という章に含まれています。

For SUSE trademarks, see https://www.suse.com/company/legal/ . All third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

訳: SUSE 社の商標については、https://www.suse.com/company/legal/ をご覧ください。その他の商標は各所有者の所有物です。商標シンボル (®, ™ など) は、それぞれ SUSE 社およびその関連会社の商標であることを示しています。また、アスタリスク (*) は第三者の商標を示しています。

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

訳: この書籍内にある全ての情報は、細部に至るまで最大限の注意を払って制作されていますが、完全に正確であることを保証するものではありません。SUSE LLC やその関連会社, 著者, 翻訳者のいずれも、本書籍内の誤りとそこから生じる結果について、一切の保証はいたしません。



注記

なお、本文書は原文 (英語) の翻訳文書であり、公式な文書ではありません。あらかじめご了承ください。

目次

前書き xvii
1 利用可能なドキュメンテーション xvii
2 ドキュメンテーションの改善 xvii
3 文書規約 xviii

I 概要 1

1 仮想化技術 2
1.1 概要 2
1.2 仮想化の利点 2
1.3 仮想化モード 3
1.4 I/O の仮想化 4

2 仮想化シナリオ 6
2.1 サーバの集約 6
2.2 分離 7
2.3 災害対策 8
2.4 動的な負荷分散 8

3 Xen 仮想化の紹介 9
3.1 基本的なコンポーネント 9
3.2 Xen 仮想化アーキテクチャ 10

4 KVM 仮想化の紹介 12
4.1 基本的なコンポーネント 12
4.2 KVM 仮想化技術 12

5 仮想化ツール 14
5.1 仮想化コンソールツール 14
5.2 仮想化 GUI ツール 15
6 仮想化コンポーネントのインストール 18
6.1 概要 18
6.2 仮想化コンポーネントのインストール 18
システムの役割の指定 18 • YaST 仮想化 モジュールの実行 19 • 特定のインストールパターンの選択 21
6.3 KVM での入れ子型仮想化 (nested virtualization) の有効化 22
VMware ESX のゲストハイパーバイザとしての使用 23
II libvirt を利用した仮想マシンの管理 25
7 libvirt デーモン 26
7.1 モジュール型デーモンの開始と停止 26
7.2 モノリシック型デーモンの開始と停止 29
7.3 モノリシック型デーモンへの切り替え 31
8 VM ホストサーバ の準備 32
8.1 ネットワークの設定 32
ネットワークブリッジ 32 • 仮想ネットワーク 37
8.2 ストレージプールの設定 47
virsh を利用したストレージの管理 50 • 仮想マシンマネージャ を利用したストレージの管理 55
9 ゲストのインストール 62
9.1 GUI ベースのゲストインストール 62
PXE 起動向けの仮想マシンの設定 65
9.2 virt-install によるコマンドラインからのインストール 66
9.3 高度なゲストインストール手順 69
高度な UEFI 設定 69 • インストール時のアドオン製品の取り込み 72

10 基本的な VM ゲスト の管理 73

10.1 VM ゲスト の一覧表示 73
仮想マシンマネージャ を利用した VM ゲスト の一覧表示 73 • virsh による VM ゲスト の一覧表示 74

10.2 コンソール経由での VM ゲスト へのアクセス 74
グラフィカルコンソールの表示 74 • シリアルコンソールへの接続 76

10.3 VM ゲスト の状態変更 (開始／停止／一時停止) 77
仮想マシンマネージャ を利用した VM ゲスト の状態変更 78 • virsh を利用した VM ゲスト の状態変更 79

10.4 VM ゲスト の状態保存と状態復元 80
仮想マシンマネージャ を利用した状態保存と状態復元 82 • virsh による状態保存と状態復元 82

10.5 スナップショットの作成と管理 83
用語 84 • 仮想マシンマネージャ を利用したスナップショットの作成と管理 84 • virsh を利用したスナップショットの作成と管理 86

10.6 VM ゲスト の削除 88
仮想マシンマネージャ を利用した VM ゲスト の削除 89 • virsh を利用した VM ゲスト の削除 89

10.7 監視 89
仮想マシンマネージャ を利用した監視 89 • virt-top を利用した監視 90 • kvm_stat を利用した監視 91

11 接続と認可 93

11.1 認証 93
libvirtd の認証 94 • VNC 認証 98

11.2 VM ホストサーバ への接続 102
非特権ユーザに対する「システム」アクセス 103 • 仮想マシンマネージャ による接続の管理 104

11.3 リモート接続の設定 105
SSH によるリモートトンネル (qemu+ssh もしくは xen+ssh) 106 • x509 証明書を利用したリモート TLS/SSL 接続 (qemu+tls もしくは xen+tls) 106

12 高度なストレージ設定 114
12.1 virtlockd を利用したディスクファイルやブロックデバイスのロック (施錠) 114
ロックの有効化 114 • ロックの設定 115
12.2 ゲストのブロックデバイスのオンラインサイズ変更 116
12.3 ホストとゲストの間でのディレクトリ共有 (ファイルシステムのパススルー) 117
12.4 libvirt を利用した RADOS ブロックデバイスの使用 118
13 仮想マシンマネージャ を利用した仮想マシンの設定 119
13.1 マシンの設定 120
概要 120 • 性能 121 • CPU 数 122 • メモリ 123 • ブートオプション 125
13.2 ストレージ 126
13.3 コントローラ 127
13.4 ネットワーク 128
13.5 入力デバイス 130
13.6 ビデオ 131
13.7 USB リダイレクト 132
13.8 その他 133
13.9 仮想マシンマネージャ を利用した CD/DVD-ROM デバイスの追加 133
13.10 仮想マシンマネージャ を利用したフロッピィデバイスの追加 134
13.11 仮想マシンマネージャ を利用したフロッピィもしくは CD/DVD-ROM メディアの取り出しと交換 135
13.12 VM ゲスト に対するホスト側の PCI デバイスの割り当て 136
仮想マシンマネージャ を利用した PCI デバイスの追加 136
13.13 VM ゲスト に対するホスト側の USB デバイスの割り当て 138
仮想マシンマネージャ を利用した USB デバイスの追加 138

14 `virsh` を利用した仮想マシンの設定 140
14.1 VM の設定変更 140
14.2 マシンの種類の変更 141
14.3 ハイパーバイザ機能の設定 142
14.4 起動オプションの変更 143
起動順序の変更 143 • 直接カーネル起動の使用 144
14.5 CPU の設定 144
CPU 数の設定 145 • CPU モデルの設定 146
14.6 メモリ割り当ての設定 147
14.7 大規模 NUMA システムにおける CPU とメモリの設定と配置 149
huge page 149 • VM ゲスト に対する自動 NUMA 配置 150
14.8 PCI デバイスの追加 151
IBM Z 向けの PCI パススルー 154
14.9 USB デバイスの追加 155
14.10 SR-IOV デバイスの追加 156
要件 156 • SR-IOV ホストドライバの読み込みと設定 157 • VM ゲスト に対する VF ネットワークデバイスの追加 160 • プールからの動的な VF の割り当て 163
14.11 接続されているデバイスの一覧表示 164
14.12 ストレージデバイスの設定 165
14.13 コントローラデバイスの設定 166
14.14 ビデオデバイスの設定 168
VRAM の割当量の変更 168 • 2D/3D アクセラレーションの設定変更 168
14.15 ネットワークデバイスの設定 169
マルチキュー型の virtio-net によるネットワーク性能の強化 169
14.16 VM ホストサーバ のネットワークインターフェイスを共有するための macvtap の使用 169
14.17 メモリバルーンデバイスの無効化 171

14.18 マルチモニタ (デュアルヘッド) の設定 171

14.19 IBM Z における KVM ゲストへの暗号化アダプタのパススルー 173
概要 173 • カバーされる範囲 173 • 要件 173 • KVM ホスト側での暗号化ドライバの占有設定 173 • さらに詳しい情報 175

15 AMD SEV-SNP による仮想マシンのセキュリティ強化 177

15.1 対応するハードウエア 177

15.2 システムの基礎部分の設定 178

15.3 設定の確認 178

15.4 AMD SEV-SNP 仮想マシンのインストール 179

15.5 AMD SEV-SNP 仮想マシンの確認 180

15.6 AMD SEV-SNP 仮想マシンの認証 180
認証レポートの生成と検証 181 • ホスト側での AMD 証明書の検証 181

15.7 現時点での制限事項 182

16 VM ゲスト の移行 184

16.1 移行の種類 184

16.2 移行における要件 185

16.3 仮想マシンマネージャ によるライブマイグレーション 186

16.4 `virsh` による移行 187

16.5 手順例 190
ストレージのエクスポート 190 • 移行先ホストでのプールの設定 190 • ボリュームの作成 192 • VM ゲスト の作成 192 • VM ゲスト の移行 192

17 Xen から KVM への移行ガイド 193

17.1 `virt-v2v` を使用した KVM への移行 193
`virt-v2v` の紹介 194 • `virt-v2v` のインストール 194 • `libvirt` 管理下の KVM 仮想マシンへの変換 194 • 変換した仮想マシンの起動 199

17.2 Xen から KVM への手動移行 200
概要 200 • Xen VM ゲスト のバックアップ 200 • 準仮想化ゲストに固有の変更 201 • Xen VM ゲスト 設定の更新 204 • VM ゲスト の移行 209

17.3 さらなる情報 209

III 全ハイパーバイザ共通の機能 210

18 ディスクのキャッシュモード 211
18.1 ディスクキャッシュとは 211

18.2 ディスクキャッシュの動作 211

18.3 ディスクキャッシュの利点 212

18.4 ディスクのキャッシュモード 212

18.5 キャッシュモードとデータの一貫性 213

18.6 キャッシュモードとライブマイグレーションの関係 213

19 VM ゲスト の時刻設定 215
19.1 KVM: kvm_clock を使用する方法 215
その他の時刻維持方式 216

19.2 Xen 仮想マシン時計設定 216

20 libguestfs 217
20.1 VM ゲスト のディスク編集の概要 217
VM ゲスト のディスク編集によるリスク 217 • libguestfs の設計 218

20.2 パッケージのインストール 219

20.3 guestfs ツール 219
仮想マシンの修正 219 • 対応するファイルシステムとディスクイメージ 220 • virt-rescue 220 • virt-resize 221 • その他の virt-* ツール 222 • guestfish 225 • 物理イメージから KVM ゲストへの変換 226

20.4 トラブルシューティング 228
btrfs 関連の問題 228 • 環境 228 • libguestfs-test-tool 229

20.5 さらなる情報 229

21 QEMU ゲストエージェント 230
21.1 QEMU GA コマンドの実行 230
21.2 QEMU GA を必要とする `virsh` のコマンド 231
21.3 `libvirt` コマンドの拡張 231
21.4 さらなる情報 232
22 ソフトウエア TPM エミュレータ 233
22.1 概要 233
22.2 事前要件 233
22.3 インストール 233
22.4 QEMU での `swtpm` の使用 233
22.5 `libvirt` での swtpm の使用 235
22.6 OVMF ファームウエアでの TPM の完全性計測 235
22.7 各種情報 235
23 VM ゲスト に対するクラッシュダンプの作成 236
23.1 概要 236
23.2 完全仮想化マシンに対するクラッシュダンプの作成 236
23.3 準仮想化マシンに対するクラッシュダンプの作成 236
23.4 追加の情報 237
IV XEN を利用した仮想マシンの管理 238
24 仮想マシンホストの設定 239
24.1 注意事項 239
24.2 Dom0 のメモリ管理 240
Dom0 のメモリ割り当ての設定 241
24.3 完全仮想化環境でのネットワークカード 242

24.4 仮想マシンホストの開始 242

24.5 PCI パススルー 244
PCI パススルー に対応するためのハイパーバイザ側の設定 245 • VM ゲスト システムに対する PCI デバイスの割り当て 246 • VGA パススルー 247 • トラブルシューティング 247 • さらなる情報 248

24.6 USB パススルー 248
USB デバイスの識別方法 248 • 擬似 USB デバイス 249 • 準仮想化 PVUSB 249

25 仮想ネットワーク 251

25.1 ゲストシステム向けのネットワークデバイス 251

25.2 Xen でのホストベースルーティング 253

25.3 マスカレード型ネットワーク設定 255

25.4 特殊な設定 258
仮想ネットワーク内での帯域制限 258 • ネットワークトラフィックの監視 258

26 仮想環境の管理 260

26.1 xl: Xen 管理ツール 260
ゲストドメインの設定ファイル 262

26.2 ゲストドメインの自動起動 262

26.3 イベントアクション 263

26.4 タイムスタンプカウンタ (TSC) 264

26.5 仮想マシンの保存 264

26.6 仮想マシンの再開 265

26.7 仮想マシンの状態 265

27 Xen 内でのブロックデバイス 267

27.1 物理ストレージから仮想ディスクへのマッピング 267

27.2 ネットワークストレージから仮想ディスクへのマッピング 268

27.3 ファイルとして存在する仮想ディスクとループバックデバイス 269

27.4 ブロックデバイスのサイズ変更 269

27.5 高度なストレージ管理向けのスクリプト 270

28 仮想化: オプション設定 271

28.1 仮想 CD リーダ 271
準仮想化マシン内での仮想 CD リーダ 271 • 完全仮想化マシン内での仮想 CD リーダ 271 • 仮想 CD リーダの追加 272 • 仮想 CD リーダの削除 273

28.2 リモートアクセス方式 273

28.3 VNC ビューア 274
仮想マシンに対する VNC ビューアのポート番号設定 275 • VNC ビューアの代替としての SDL の使用 275

28.4 仮想キーボード 275

28.5 CPU リソースの占有 276
Dom0 側の設定 276 • VM ゲスト 側の設定 277

28.6 HVM 機能 278
起動時のブートデバイス指定 278 • ゲスト向けの CPUID 変更 278 • PCI-IRQ 数の拡張 279

28.7 仮想 CPU のスケジューリング 280

29 管理作業 281

29.1 ブートローダプログラム 281

29.2 スパースイメージファイルとディスク領域 282

29.3 Xen VM ゲスト システムの移行 284
CPU 機能の検出 284 • 移行のためのブロックデバイスの準備 286 • VM ゲスト システムの移行 286

29.4 Xen の監視 287
`xentop` を利用した Xen の管理 287 • 追加のツール 288

29.5 VM ゲスト システムに対するホスト情報の提供 289

30 XenStore: ドメイン間で共有される設定データベース 290
30.1 概要 290
30.2 ファイルシステムインターフェイス 290
XenStore のコマンド 291 • /vm 292 • /local/domain/<ドメイン_ID> 294
31 Xen の高可用性仮想化ホストとしての使用 296
31.1 リモートストレージでの Xen HA 構成 296
31.2 ローカルストレージでの Xen HA 構成 297
31.3 Xen HA とプライベートブリッジ 298
32 Xen: 準仮想化 (PV) ゲストから完全仮想化 (FV/HVM) ゲストへの変換 299
V QEMU を利用した仮想マシンの管理 303
33 QEMU の概要 304
34 KVM VM ホストサーバ の構築 305
34.1 仮想化のための CPU 側サポート 305
34.2 必要なソフトウエア 305
34.3 KVM ホスト固有の機能 307
virtio-scsi を利用したホスト側ストレージの使用 307 • vhost-net を利用したネットワーク処理の高速化 308 • マルチキュー型 virtio-net を利用したネットワーク性能の強化 309 • VFIO: デバイスに対する安全な直接アクセス 310 • VirtFS: ホストとゲストの間でのディレクトリ共有 312 • KSM: ゲスト間でのメモリページ共有 313
35 ゲストのインストール 315
35.1 qemu-system-ARCH を利用した基本的なインストール 315

35.2 qemu-img を利用したディスクイメージの管理 317
qemu-img の実行に関する一般的な情報 317 • ディスクイメージの作成／変換／検査 318 • qemu-img を利用した仮想マシンのスナップショット管理 323 • ディスクイメージの効率的な使用 326

36 qemu-system-ARCH を利用した仮想マシンの実行 331

36.1 基本的な qemu-system-ARCH の実行方法 331

36.2 一般的な qemu-system-ARCH のオプション 332
基本的な仮想ハードウエア構成 334 • 仮想デバイスの設定保存と読み込み 335 • ゲスト側のリアルタイムクロック 336

36.3 QEMU 内でのデバイスの使用 337
ブロックデバイス 337 • グラフィックデバイスとディスプレイのオプション 343 • USB デバイス 345 • キャラクタデバイス 346

36.4 QEMU でのネットワーク 349
ネットワークインターフェイスカードの定義 349 • ユーザモードネットワーク 350 • ブリッジ型ネットワーク 352

36.5 VNC を利用した VM ゲスト の表示 355
VNC 接続の認証設定 357

37 QEMU モニタを利用した仮想マシンの管理 360

37.1 モニタコンソールへのアクセス 360

37.2 ゲストシステムに関する情報の取得 361

37.3 VNC パスワードの変更 364

37.4 デバイスの管理 364

37.5 キーボードとマウスの制御 365

37.6 利用可能なメモリの変更 366

37.7 仮想マシンのメモリダンプ 366

37.8 仮想コンソールのスナップショットの管理 367

37.9 仮想マシンの一時停止と再開 368

37.10 ライブマイグレーション 369

37.11 QMP - QEMU マシンプロトコル 370
標準入出力経由での QMP アクセス 370 • Telnet 経由での QMP アクセス 372 • Unix ソケット経由での QMP アクセス 372 • `libvirt` の `virsh` コマンド経由での QMP アクセス 373

VI トラブルシューティング 374

38 内蔵ヘルプとパッケージのドキュメンテーション 375

39 システム情報とログの収集 377

39.1 `libvirt` のログ制御 377

用語集 379

A NVIDIA カードに対する GPU パススルー の設定 389

A.1 概要 389

A.2 事前要件 389

A.3 ホスト側の設定 389
ホスト側の環境確認 389 • IOMMU の有効化 390 • Nouveau ドライバのブラックリスト設定 391 • VFIO の設定とパススルーのための GPU 分離 391 • VFIO ドライバの読み込み 391 • Microsoft Windows ゲストに対する MSR の無効化 392 • UEFI ファームウエアのインストール 392 • ホストマシンの再起動 393

A.4 ゲスト側の設定 393
ゲスト側の設定を行うための要件 393 • グラフィックカードドライバのインストール 394

B GNU ライセンス 397

前書き

改訂履歴

2023-02-03

1 利用可能なドキュメンテーション

オンラインドキュメンテーション

ドキュメンテーションは https://doc.opensuse.org で公開されています。ここから直接読むこともできますし、様々な形式でダウンロードを行うこともできます。



注記: 最新の更新について

ドキュメンテーションの最新版は通常、英語版が先に作成されます。

SUSE ナレッジベース

何らかの問題に直面した場合は、https://www.suse.com/support/kb/ からアクセスできる技術情報文書 (TID) をご確認ください。ここからお客様からの問い合わせで発生した、様々な SUSE 社の知識情報が検索できます。

お使いのシステム内

オフライン環境での利用を想定して、お使いのシステム内の `/usr/share/doc/release-notes` ディレクトリにはリリースノートが保存されているほか、各パッケージに対するドキュメンテーションが `/usr/share/doc` 内に存在しています。
また、多くのコマンドに対して、マニュアルページ も用意されています。マニュアルページは `man` コマンドで表示することができます。このコマンドの後ろに参照したいコマンド名を指定してください。なお、`man` コマンドがインストールされていない場合は、`sudo zypper install man` を実行してインストールしてください。

2 ドキュメンテーションの改善

このドキュメンテーションに対するご意見だけでなく、改善や追記などの貢献をいただければ幸いです。それぞれ下記のチャンネル経由でお送りいただくことができます:

バグ報告

ドキュメンテーション内に誤記などを見つけた場合は、https://bugzilla.opensuse.org/ で問題を報告してください。

なお、この文書の HTML 版の各章には、問題を報告するための［Report a bug］というアイコンが用意されていますので、こちらをお使いのうえ報告をお願いいたします。これにより、Bugzilla で対象の製品や分類、そしてリンク先などをそれぞれ自動で設定するようになっています。あとは問題点の説明を記述するだけです。
なお、報告には Bugzilla のアカウントが必要となるほか、英語でのやり取りが必要となりますので、あらかじめご了承ください。

貢献

このドキュメンテーションに対して追記もしくは更新すべき具体的な内容をお持ちの場合は、この文書の HTML 版の各章に用意されている［EDIT SOURCE］のリンクをお使いください。これにより、GitHub 内にある 英語版原文の ソースコードを表示し編集することができますので、作業が終わり次第 pull request を送信してください。
なお、GitHub のアカウントが必要となります。



注記: ［Edit Source］は英語版のみに提供される件について

［EDIT SOURCE］のリンクは各文書の英語版原文を編集する目的でのみご利用いただけます。その他の言語については［Report a bug］でお知らせください。

なお、本文書で使用しているドキュメンテーション環境の情報については、リポジトリ内の README をお読みください。

電子メール

本製品のドキュメンテーションに対するフィードバックは、`doc-team@suse.com` でも受け付けております。なお、文書のタイトルと製品のバージョン、およびドキュメンテーションの発行日付をそれぞれご記入ください。また、問題点の報告や記述の追加に関するご提案は、それぞれ概要と対応するセクション番号、およびページ (もしくは URL) をご記入ください。

ヘルプ

openSUSE Leap に対するさらなる支援をご希望の場合は、https://ja.opensuse.org/Portal:Support をご覧ください。

3 文書規約

この文書内では、下記のような記述ルールを使用しています:

- `/etc/passwd` : ディレクトリ名やファイル名を示しています
- `PLACEHOLDER` : `PLACEHOLDER` の箇所は、実際の値に置き換えるべきものであることを示しています

- `PATH` : 環境変数であることを示しています
- `ls` , `--help` : コマンドやオプション、パラメータであることを示しています
- `user` : ユーザ名やグループ名であることを示しています
- **`パッケージ名`** : ソフトウエアのパッケージ名を示しています
- `Alt` , `Alt`–`F1` : キー入力や組み合わせキー入力を示しています; キーはキーボードに書かれているとおりに大文字で示されます
- [ファイル] , [ファイル] › [名前を付けて保存] : メニュー項目やボタンなどを示しています
- 第1章「章のタイトル」: 本ガイド内の他の箇所への参照を示しています。
- 下記は `root` ユーザの権限で実行しなければならないコマンドを示しています。一般ユーザから実行する場合は、これらのコマンドの前に `sudo` を付けることで、`root` で実行できるようになります:

```
# コマンド
> sudo コマンド
```

- 下記は一般ユーザで実行できるコマンドを示しています:

```
> コマンド
```

- また、行末にバックスラッシュ文字 (`\`) を付けることで、コマンドを複数行に分けて記述している場合もあります。バックスラッシュ文字は、シェルに対して、これ以降にもコマンドが続くことを示す文字になります:

```
> echo a b \
c d
```

- このほか、行頭にプロンプトが書かれたコマンド行に続いて、そのコマンドを実行した場合の出力例を示す場合もあります:

```
> コマンド
出力
```

- 各種の情報について



警告: 警告

実際に実施したりする前に、注意しておかなければならない、きわめて重要な情報を記述しています。セキュリティ面の問題のほか、データを失ってしまう可能性への告知、ハードウエアの損傷の可能性や物理的な障害が発生する可能性を示しています。



重要: 重要な情報

実際に実施する前に注意すべき点を説明しています。



注記: 一般的な情報

一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ヒント: その他のヒント

ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

- 簡潔な補足情報



一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

I 概要

1 仮想化技術 2

2 仮想化シナリオ 6

3 Xen 仮想化の紹介 9

4 KVM 仮想化の紹介 12

5 仮想化ツール 14

6 仮想化コンポーネントのインストール 18

1 仮想化技術

改訂履歴

2024-06-27

仮想化とは、他のオペレーティングシステム (ゲスト仮想マシン) をホストのオペレーティングシステム (ホスト) の内側で動作させることのできる技術を意味します。

1.1 概要

openSUSE Leap には Xen および KVM と呼ばれる、最新のオープンソース仮想化技術が含まれています。これらはハイパーバイザとも呼ばれ、openSUSE Leap では、単一の物理システム内に複数の仮想マシン (VM ゲスト) を配置／廃止／インストール／監視／管理することができます (詳しくは ハイパーバイザ をお読みください) 。また openSUSE Leap では、仮想化向けに高度なチューニングを施した準仮想化オペレーティングシステムのマシンと、何も修正していない完全仮想化オペレーティングシステムのマシンの両方を作成することができます。

仮想化を有効化するオペレーティングシステムで、最も重要なコンポーネントはハイパーバイザ (もしくは仮想マシンマネージャ) です。これはサーバのハードウエア内で直接動作するソフトウエアで、プラットフォームのリソースを制御したり、VM ゲスト や VM ゲスト 内で動作するオペレーティングシステムの間で、仮想化されたハードウエアインターフェイスを提示して、リソースを共有したりするための仕組みです。

openSUSE は Linux サーバオペレーティングシステムであり、2 種類のハイパーバイザ (Xen および KVM) に対応しています。

さらに Xen や KVM を動作させた openSUSE Leap は、仮想化ホストサーバ (VHS, VHS) としても動作させることができます。これは VM ゲスト 側で、独自のゲストオペレーティングシステムを動作させることができるものです。また、SUSE の VM ゲスト アーキテクチャには、多数のアプリケーションをホスティングする VM ゲスト を動作させることのできる、VHS 向けのハイパーバイザと管理コンポーネントが含まれています。

Xen では管理コンポーネントを特権下の VM ゲスト 内で動作させる仕組みであり、これはしばしば Dom0 と呼ばれます。KVM では Linux カーネルがハイパーバイザとして動作し、管理コンポーネントは VHS 内で直接動作します。

1.2 仮想化の利点

仮想化技術を使用することで、物理的なサーバでのサービス提供と同じ状態を維持したまま、さまざまな利点を得ることができるようになります。

まずはインフラストラクチャのコスト削減があげられます。サーバは主にユーザに対してサービスを提供するために使用するものですが、仮想化されたオペレーティングシステムを使用することで、同じサービス提供を維持したまま下記を実現することができます:

- メンテナンスコストの削減: 1 つのホストで複数のオペレーティングシステムを動作させることができますので、ハードウエアのメンテナンスにかかるコストも減らすことができます。
- 電力コスト／空調コストの削減: ハードウエアを 1 台にまとめることができることから、省電力に繋がるだけでなく、予備電力の用意や空調の用意などを、より少なくすることができるようになります。
- 占拠空間の削減: サーバの台数を減らすことができれば、データセンター内の占拠空間も削減することができます。
- 管理コストの削減: VM ゲスト を使用することで、インフラストラクチャの管理にかかる手間も減らすことができます。
- 運用性と生産性の改善: 仮想化には 移行 移行だけでなく、ライブマイグレーション や スナップショット 機能も含まれています。これらの機能を使用することで、サーバの停止時間を減らすだけでなく、サービスの停止を伴うことなく一方から他方にサービスを移動させることができるようになります。

1.3 仮想化モード

仮想マシン内でゲストオペレーティングシステムを動作させる際、完全仮想化 (FV) モードと準仮想化 (PV) モードのいずれかを選択することができます。それぞれの仮想化モードには、それぞれ利点と欠点が存在しています。

- 完全仮想化モードでは、Windows* Server など、何も変更していないオペレーティングシステムをそのまま仮想マシンとして動作させることができます。バイナリ変換を行うか、もしくは AMD* Virtualization や Intel* Virtualization Technology などのハードウエア支援型 (ハードウエア支援) 仮想化技術のいずれかを使用します。ハードウエア支援型の仕組みは、対応するプロセッサを使用することで、よりよい性能を提供できる仕組みです。
- 準仮想化モードで動作させる場合、一般にゲスト側のオペレーティングシステムを仮想化環境に合わせて修正する必要があります。その代わり、完全仮想化でオペレーティングシステムを動作させる場合に比べて準仮想化は、より高速にオペレーティングシステムを動作させることができます。
  準仮想化に対応するよう修正されたオペレーティングシステムは、準仮想化対応済みオペレーティングシステム と呼ばれ、openSUSE Leap がそれに該当します。

1.4 I/O の仮想化

VM ゲスト はホストシステム内の CPU やメモリリソースを共有するだけでなく、I/O サブシステムを共有することもできます。ソフトウエアによる I/O 仮想化技術はベアメタル (物理的なサーバ) に比べると性能が落ちますが、ハードウエアによる技術を使用すると、ほぼ「ネイティブ」に近い性能を発揮できるようになっています。openSUSE Leap では、下記のような I/O 仮想化技術に対応しています:

完全仮想化

完全仮想化 (Fully Virtualized (FV)) ドライバでは、幅広く使用されている実在デバイスを擬似する仕組みであるため、VM ゲスト 内でも既存のドライバを使用することができます。ゲストはハードウエア仮想マシン (Hardware Virtual Machine; HVM) と呼ぶこともあります。VM ホストサーバ 内の物理デバイスは擬似されているデバイスとは異なるため、ハイパーバイザはゲスト側からの全ての I/O 操作を物理デバイスに渡す際、それらを正しく処理しなければならないことを意味します。そのため、全ての I/O 操作はソフトウエア層にまで引き上げて処理を行うことになりますので、I/O 性能がかなり落ちてしまうだけでなく、CPU 側にも負荷がかかることになります。

準仮想化

準仮想化 (Paravirtualization (PV)) ではハイパーバイザと VM ゲスト の間で直接的な通信を行います。オーバーヘッドの少ない仕組みであることから、性能は完全仮想化に比べてずっと良好になります。しかしながら、準仮想化ではゲスト側のオペレーティングシステムを修正し、準仮想化 API に対応させるようにするか、準仮想化用のドライバを使用する必要があります。

PVHVM

この種類の仮想化では、HVM (完全仮想化 をお読みください) でありながら、準仮想化 (PV) ドライバと PV による割り込みやタイマー処理を使用します。

VFIO

VFIO は Virtual Function I/O の略で、Linux 向けの新しいユーザレベルドライバのフレームワークです。従来の KVM PCI パススルー デバイス割り当てを置き換えるものです。VFIO ドライバはユーザスペースに対して、メモリの機密を保持 (IOMMU) しながら、保護環境下で直接のデバイスアクセス機能を提供するものです。VFIO を利用することで、VM ゲスト は VM ホストサーバ 内のハードウエアデバイスに、性能面のクリティカルパスの擬似による性能劣化を引き起こすことなく、直接アクセスできるようになります。ただし、この方式はデバイスの共有には使用できません。各デバイスは単一の VM ゲスト にのみ割り当てることができます。また、VFIO は VM ホストサーバ の CPU やチップセット、BIOS/EFI でそれぞれ対応している必要があります。

従来型の KVM PCI デバイス割り当てと比較して、VFIO には下記のような利点があります:

- リソースへのアクセスは UEFI Secure Boot との互換性があります。
- デバイスは孤立した存在になり、メモリアクセスも保護されます。
- より柔軟なデバイス所有者モデルを採用した、ユーザスペースのデバイスドライバが提供されています。
- KVM 技術とは独立しているほか、x86 アーキテクチャのみに固有のものでもありません。

openSUSE Leap では、USB と PCI のパススルーによるデバイスの割り当ては廃止予定とされ、VFIO モデルで置き換えられています。

SR-IOV

最新の I/O 仮想化技術で、Single Root I/O Virtualization の略です。SR-IOV では前述の技術の利点を組み合わせて作られています。性能を強化しているだけでなく、複数の VM ゲスト でデバイスを共有する機能を備えています。SR-IOV では、リソースを複製して複数のデバイスとしてアクセスすることのできる、特殊な I/O デバイスが必要となります。この場合、それぞれの「擬似」デバイスは 1 台のゲストからアクセスすることになります。しかしながら、たとえばネットワークカードなどの場合、同時に使用することのできるキュー数は制限されているため、準仮想化に比べると VM ゲスト の性能が落ちる可能性があることになります。VM ホストサーバ 側としては、SR-IOV が I/O デバイス側と CPU 、チップセットと BIOS/EFI 、そしてハイパーバイザでそれぞれ対応していなければなりません。詳しい設定手順については、13.12項「VM ゲストに対するホスト側の PCI デバイスの割り当て」をお読みください。



重要: VFIO および SR-IOV の要件について

VFIO や SR-IOV の機能を使用できるようにするには、VM ホストサーバ 側が下記の要件を満たす必要があります:

- BIOS や EFI で IOMMU が有効化されていること。
- Intel CPU の場合、カーネルのコマンドライン内のパラメータに `intel_iommu=on` を追加する必要があります。詳しくは https://github.com/torvalds/linux/blob/master/Documentation/admin-guide/kernel-parameters.txt#L1951 (英語) をお読みください。
- VFIO インフラストラクチャを利用できるようにする必要があります。こちらはカーネルモジュールである `vfio_pci` を読み込むことで、利用できるようになります。詳しくは『リファレンス』、第10章「`systemd` デーモン」、10.6.4項「カーネルモジュールの読み込み」をお読みください。

2 仮想化シナリオ

改訂履歴

2023-12-01

仮想化の仕組みは、たとえば下記のようなメリットをもたらします:

- より効率的なハードウエアの使用
- 古いソフトウエアのサポート
- オペレーティングシステムの分離
- ライブマイグレーション
- 災害対策
- 負荷分散

2.1 サーバの集約

多数のサーバを 1 台の巨大な物理サーバに統合することができます。多数のサーバを仮想マシンに変換することで、ハードウエアを集約してまとめることができます。古いソフトウエアを新しいハードウエア上で動作させることもできます。

- リソースを効率的に使用することができます (一般的には各サーバが全てのリソースを使用しているわけではないため)
- サーバの物理スペースを削減することができます
- 複数の処理を同じサーバ内で処理することにより、効率の向上を図ることができます
- データセンターのインフラストラクチャを単純化することができます
- 他のホストへの負荷移行を簡単に行うことができます (サービスのダウンタイム防止)
- 手っ取り早くマシンを調達するための手段としても使用することができます
- 単一のホスト内で複数のゲストオペレーティングシステムを動作させることができます



重要

サーバの統合にあたっては、下記の点について特別な注意を払う必要があります:

- メンテナンス時間を注意深く計画する必要があります
- ストレージが統合の鍵となります。移行に対応する必要があるほか、ディスク領域の拡張にも対応する必要があります
- さらなる負荷をまかなえるかどうかをよく確認しておく必要があります

2.2 分離

ゲスト側のオペレーティングシステムはホスト側のオペレーティングシステムとの間で完全に分離されています。そのため、仮想マシン内で何らかの問題が発生した場合でも、ホスト側はその影響を受けません。また、一方の仮想マシンで問題が発生しても、他方の仮想マシンに影響を与えることもありません。仮想マシン間でのデータの共有も行われません。

- 仮想マシンに対して UEFI Secure Boot を使用することができます。
- KSM は使用すべきではありません。KSM に関する詳細は KSM をお読みください。
- CPU の各コアはそれぞれの仮想マシン専用にすることもできます。
- 潜在的なセキュリティ問題を避けるため、Hyper-threading (HT) 機能は無効化しておくことをお勧めします。
- 仮想マシンはネットワーク／ストレージ／ハードウエアを共有すべきではありません。
- PCI パススルー機能や NUMA などの高度なハイパーバイザ機能を使用する場合は、仮想マシンの移行時に問題が生じうることに注意しておいてください
- 一般的に、準仮想化ドライバ `virtio` を使用することで、仮想マシンの性能と効率の両方を向上させることができます

AMD 社は仮想化のセキュリティに関する特別な機能を提供しています。

2.3 災害対策

ハイパーバイザにはスナップショット機能が用意されていますので、サーバを任意の時点に巻き戻すことができます。仮想化ゲスト OS は物理サーバで動作する場合とは異なり、実際のハードウエアから独立した存在になっていますので、スナップショットを同じハイパーバイザの動作する別のマシン内に復元すれば、そのままサーバを移行することができます。

2.4 動的な負荷分散

ライブマイグレーションはお使いのインフラストラクチャの負荷分散を行うための単純な方法です。仮想マシンを負荷の高いホストから負荷の少ないホストに移動させることで、容易に負荷を調整することができます。

3 Xen 仮想化の紹介

改訂履歴

2024-06-27

本章では、Xen ベースの仮想化環境を構築したり管理したりするにあたって、あらかじめ理解しておくべきコンポーネントや技術を、紹介および説明しています。

3.1 基本的なコンポーネント

Xen ベースの仮想化環境は、下記のようなコンポーネントから構成されます:

- Xen ハイパーバイザ
- Dom0
- 任意の数の VM ゲスト
- 仮想化を管理するためのツール／コマンド／設定ファイル

また、これらのコンポーネントを動作させている物理コンピュータを総称して、VM ホストサーバ と呼びます。これは、仮想マシンの提供者 (ホスト) を構成しているためです。

Xen ハイパーバイザ

Xen ハイパーバイザは、仮想マシンモニタと呼ばれることもありますが、仮想マシンと物理ハードウエアとの間を、低レベルな (ハードウエアに近い) 範囲で仲介するオープンソースのソフトウエアプログラムです。

Dom0

仮想マシンホストの環境で、Dom0 や制御ドメインと呼ばれます。Dom0 は下記のようなコンポーネントから構成されています:

- openSUSE Leap では、仮想マシンホストのコンポーネントや仮想マシン自体を管理するためのグラフィカルおよびコマンドラインの環境が提供されています。



注記

「Dom0」の用語にあるとおり、これは管理環境を提供する特殊なドメインです。グラフィカル環境でもコマンドラインモードでも動作させることができます。

- xenlight ライブラリ (libxl) をベースにした xl ツールスタック。Xen のゲストドメインを管理する際に使用します。
- QEMU は完全なコンピュータシステムを擬似するオープンソースソフトウエアで、プロセッサやさまざまな周辺機器を擬似することができます。オペレーティングシステムを完全仮想化もしくは準仮想化のいずれかで動作させる機能を提供します。

Xen ベースの仮想マシン

Xen ベースの仮想マシンは `VM ゲスト` や `DomU` と呼ばれることもありますが、これは下記のコンポーネントから構成されるものです:

- 起動可能なオペレーティングシステムを含む 1 台もしくは複数台の仮想ディスク。仮想ディスクはファイルベースのほか、パーティションベースやボリュームベース、もしくはその他のブロックデバイスをベースにすることもできます。
- ゲストドメインごとの設定ファイル。これはテキストファイル形式で、`man 5 xl.conf` で表示することのできるマニュアルページで書式が説明されているものです。
- 1 つもしくは複数のネットワークデバイス。制御ドメインが提供する仮想ネットワークに接続します。

管理ツール／コマンド／設定ファイル

GUI ツールやコマンド、設定ファイルなどの仕組みにより、お使いの仮想環境を管理したりカスタマイズしたりすることができます。

3.2 Xen 仮想化アーキテクチャ

下記の図は仮想マシンホストの中に 4 台の仮想マシンが存在する場合の例です。Xen ハイパーバイザ (Hypervisor) は物理ハードウエアプラットフォーム (Physical Machine) 上で直接動作しています。なお、制御ドメイン (Virtual Machine Host Server) も仮想マシンそのものですが、通常の仮想マシンの機能に加えて、管理作業を実施できる機能が与えられていることに注意してください。

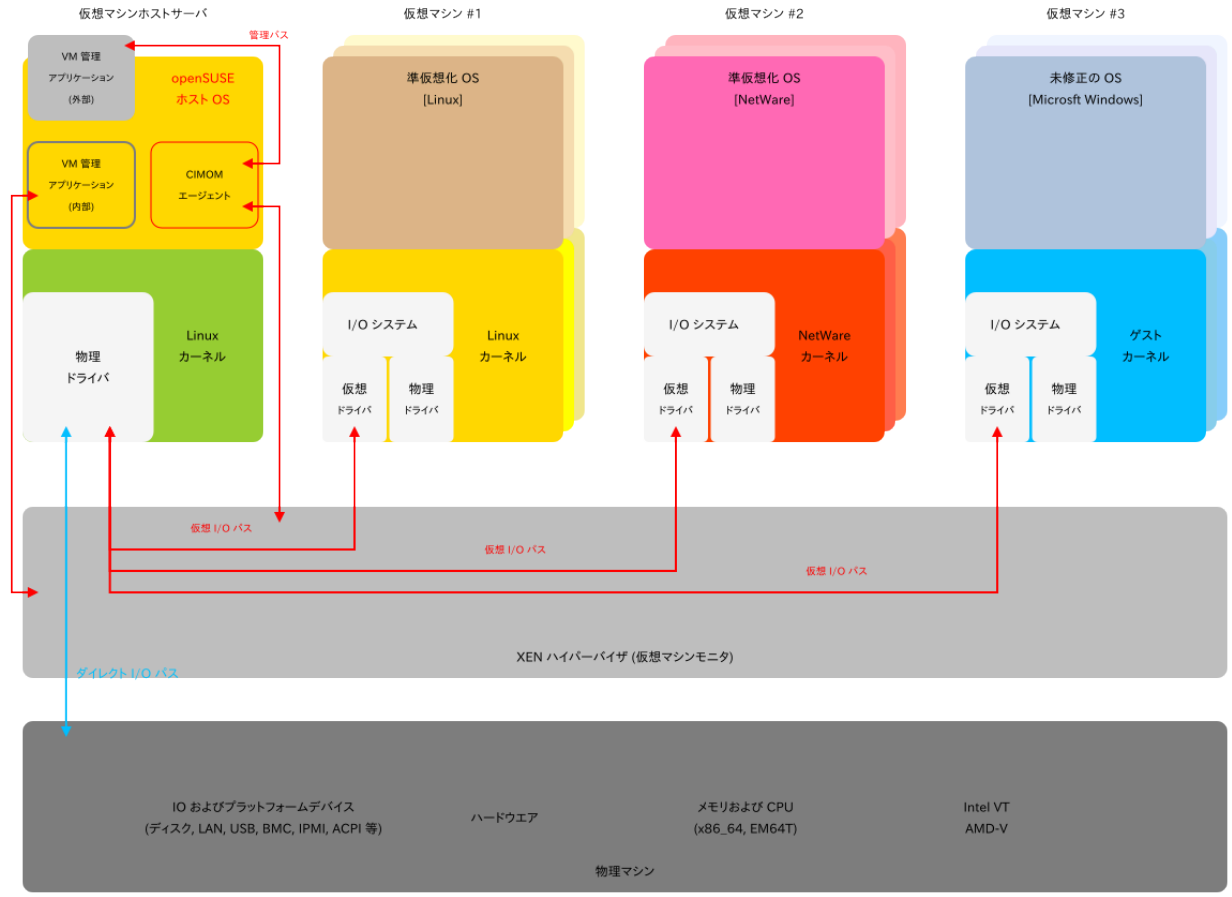


図 3.1: XEN 仮想化アーキテクチャ

一番左には Dom0 の仮想マシンホストが描かれ、こちらでは openSUSE Leap オペレーティングシステムを動作させているものとします。真ん中の 2 台の仮想マシンは準仮想化 (Paravirtualized) の OS を動作させていて、右側では何も変更を加えていない (Unmodified) オペレーティングシステム (Microsoft Windows や Microsoft Windows Server など) を動作させています。

4 KVM 仮想化の紹介

改訂履歴

2023-06-06

4.1 基本的なコンポーネント

KVM はハードウエア仮想化機能を持つアーキテクチャ向けの完全仮想化ソリューションです 。

VM ゲスト (仮想マシン), 仮想ストレージ, 仮想ネットワークの管理は、QEMU のツールから直接実行することができるほか、`libvirt` ベースのスタックでも管理することができます。QEMU ツールには `qemu-system-ARCH` が含まれているほか、QEMU モニタや `qemu-img` , `qemu-ndb` なども含まれています。また、`libvirt` ベースのスタックには `libvirt` それ自身のほか、`virsh` , `virt-manager` , `virt-install` , `virt-viewer` などの `libvirt` ベースのアプリケーションが含まれています。

4.2 KVM 仮想化技術

この完全仮想化型のソリューションは、主に 2 つのコンポーネントから構成されています:

- 仮想化の中枢インフラストラクチャと、プロセッサ別のドライバであるカーネルモジュール (`kvm.ko` , `kvm-intel.ko` , `kvm-amd.ko`) 。
- VM ゲスト (仮想マシン) における仮想デバイスのエミュレーションや制御の仕組みを提供するユーザスペースプログラム (`qemu-system-アーキテクチャ`) 。

KVM という用語は、狭い意味ではカーネルレベルの仮想化機能を指す用語ですが、実際にはユーザスペースのコンポーネントを含む用語として使用しています。

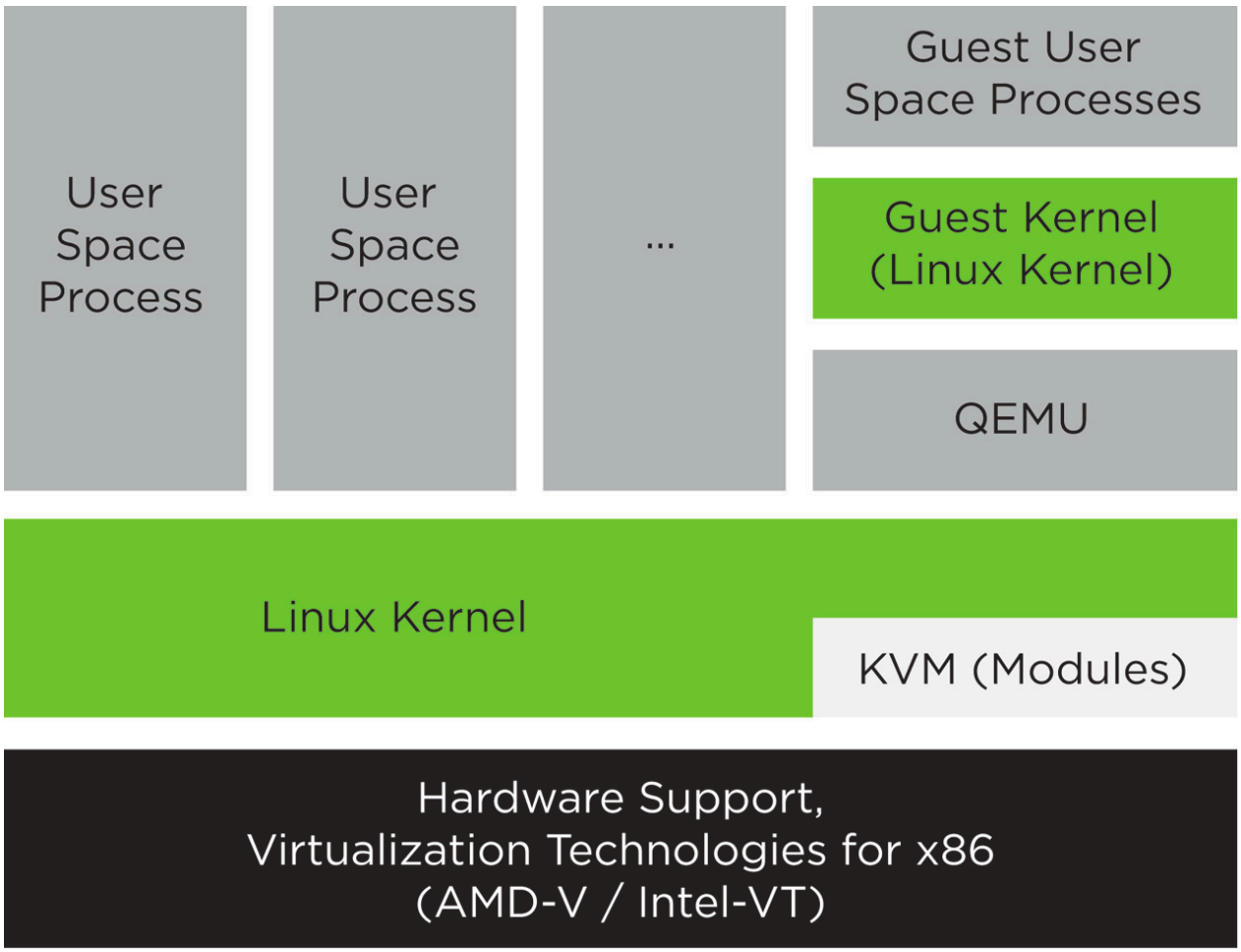


図 4.1: KVM 仮想化技術

5 仮想化ツール

改訂履歴

2024-06-27

`libvirt` は KVM や Xen など、著名な仮想化ソリューションに対応した管理機能の汎用 API を提供するライブラリです。このライブラリはこれらの仮想化ソリューションに対して統一された管理 API を提供し、高レベルな管理ツール向けに安定したハイパーバイザに依存しないインターフェイスを提供しています。このライブラリには、VM ホストサーバ における仮想ネットワークや仮想ストレージの管理機能も用意されています。また、各 VM ゲスト の設定は、XML ファイル内に保存される仕組みでもあります。

`libvirt` を使用することで、VM ゲスト をリモートから管理することもできます。TLS による暗号化のほか、x509 形式の証明書や、SASL による認証にも対応しています。このような構造により、単一のワークステーションから複数の VM ホストサーバ を一括管理することができますので、VM ホストサーバ を個別に管理する必要性を削減することができます。

また、`libvirt` ベースのツールは VM ゲスト を管理する際の推奨される方法です。`libvirt` と `libvirt` ベースのアプリケーションとの間の相互運用性はテスト済みで、SUSE のサポートを受ける際にも推奨される方法でもあります。

5.1 仮想化コンソールツール

`libvirt` には、仮想マシンを管理するためのコマンドラインユーティリティがいくつか用意されています。主なものは下記のとおりです:

`virsh` (パッケージ名: `libvirt-client`)

仮想マシンマネージャ と同様の機能を持つ VM ゲスト の管理向けのコマンドラインツールです。VM ゲスト の状態を変更することができる (開始, 停止, 一時停止) ほか、新しいゲストやデバイスを設定したり、既存の設定を編集したりすることができます。また、`virsh` を利用することで、VM ゲスト の管理操作をスクリプト化する際にも便利です。

`virsh` のパラメータには最初にコマンドを、それ以降にコマンドに対応したパラメータを記述します:

```
virsh [-c URI] コマンド ドメイン_ID [オプション]
```

`zypper` と同様に、`virsh` もコマンド無しで実行することができます。この場合、専用のシェルが起動され、コマンドを待ち受ける状態になります。このモードは、複数のコマンドを実行するような場合に便利です:

```
~> virsh -c qemu+ssh://wilber@mercury.example.com/system
Enter passphrase for key '/home/wilber/.ssh/id_rsa':
Welcome to virsh, the virtualization interactive terminal.

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

virsh # hostname
mercury.example.com
```

`virt-install` (パッケージ名: `virt-install`)

`libvirt` ライブラリを利用して新しい VM ゲストを作成するためのコマンドラインツールです。VNC プロトコルや SPICE プロトコルを利用した、グラフィカルなインストールにも対応しています。また、`virt-install` では、適切なコマンドラインパラメータを指定することで、完全に無人の環境でも動作させることができます。これにより、ゲストのインストールを簡単に自動化することができます。`virt-install` は、仮想マシンマネージャで使用される既定のインストールツールでもあります。

`remote-viewer` (パッケージ名: `virt-viewer`)

リモートデスクトップ向けのシンプルなビューアです。SPICE, VNC の各プロトコルに対応しています。

`virt-clone` (パッケージ名: `virt-install`)

`libvirt` ハイパーバイザ管理ライブラリを利用して、既に存在する仮想マシンのイメージを複製するためのツールです。

`virt-host-validate` (パッケージ名: `libvirt-client`)

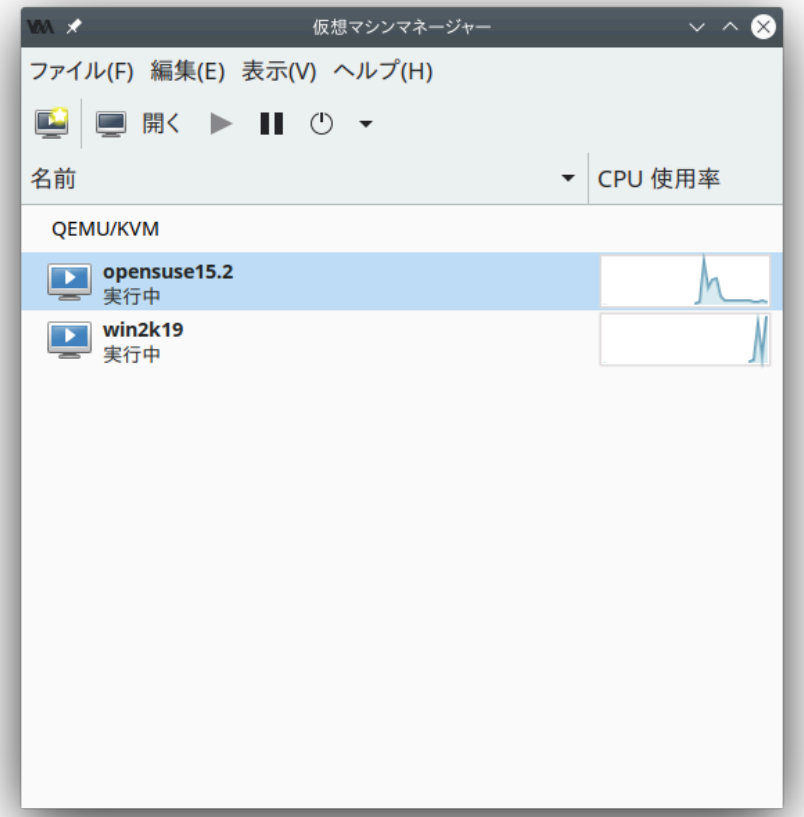
ホスト側が、`libvirt` のハイパーバイザドライバを適切に使用するように設定されているかどうかを検証するツールです。

5.2 仮想化 GUI ツール

openSUSE Leap で利用できるグラフィカルな libvirt ベースのツールには、下記のようなものがあります。

仮想マシンマネージャ (パッケージ名: `virt-manager`)

仮想マシンマネージャ は VM ゲスト を管理するためのデスクトップツールです。既存のマシンのライフサイクル制御 (起動, 停止, 一時停止, 再開, 保存, 復元) のほか、新しい VM ゲスト の作成にも使用することができます。また、さまざまな種類のストレージや仮想ネットワークを管理することもできるほか、内蔵の VNC ビューアを介して、VM ゲスト のグラフィカルコンソールに接続する機能も備えています。また、性能に関わる統計情報も取得することができます。`virt-manager` では、ローカルの `libvirtd` の管理だけでなく、`libvirtd` が動作しているリモートの VM ホストサーバ を管理することもできます。



仮想マシンマネージャ を起動するには、コマンドプロンプトから `virt-manager` と入力して実行してください。

注記

spice を使用している場合、USB デバイスの自動転送機能を無効化したい場合は、`virt-manager` を起動する際に `--spice-disable-auto-usbredir` オプションを設定するか、もしくは下記のコマンドを入力して実行し、恒久的に設定を適用してください:

```
> dconf write /org/virt-manager/virt-manager/console/auto-redirect false
```

`virt-viewer` (パッケージ名: `virt-viewer`)

VM ゲストのグラフィカルコンソールを閲覧するためのビューアです。SPICE (VM ゲストでは既定で設定されています) もしくは VNC プロトコルを利用することができるほか、TLS や x509 証明書にも対応しています。VM ゲストは名前や ID, UUID でアクセスすることができます。その時点でゲストが動作していない場合、ビューアが接続を試す前に、ゲストが起動するまで待機させることもできます。なお、`virt-viewer` は既定ではインストールされていませんので、`virt-viewer` パッケージをインストールしてお使いください。

注記

spice を使用している場合、USB デバイスの自動転送機能を無効化したい場合は、`--spice-usbredir-auto-redirect-filter=''` オプションを指定して、空のフィルタを追加してください。

`yast2 vm` (パッケージ名: `yast2-vm`)

仮想化ツールのインストールやネットワークブリッジの設定を単純化することのできる、YaST のモジュールです:



6 仮想化コンポーネントのインストール

改訂履歴

2024-06-27

6.1 概要

1 つまたは複数のゲストシステム (VM ゲスト) を実行する仮想化サーバ (VM ホストサーバ) を動作させるには、サーバ内に仮想化コンポーネントをインストールする必要があります。インストールすべきコンポーネントは、利用したい仮想化技術によって異なります。

6.2 仮想化コンポーネントのインストール

VM ホストサーバ を実行するのに必要な仮想化ツールをインストールするには、下記のいずれかの方式をとることができます:

- VM ホストサーバ 側での openSUSE Leap のインストール時に、システムの役割を選択する方式
- 既にインストール済みで動作中の openSUSE Leap において、YaST 仮想化 モジュールを実行する方式
- 既にインストール済みで動作中の openSUSE Leap において、特定のパターンを選択してインストールする方式

6.2.1 システムの役割の指定

VM ホストサーバ 側での openSUSE Leap のインストールの際、仮想化に必要なツール類全てをインストールするよう選択することができます。具体的にはインストールの際、［システムの役割］画面でそれを行います。

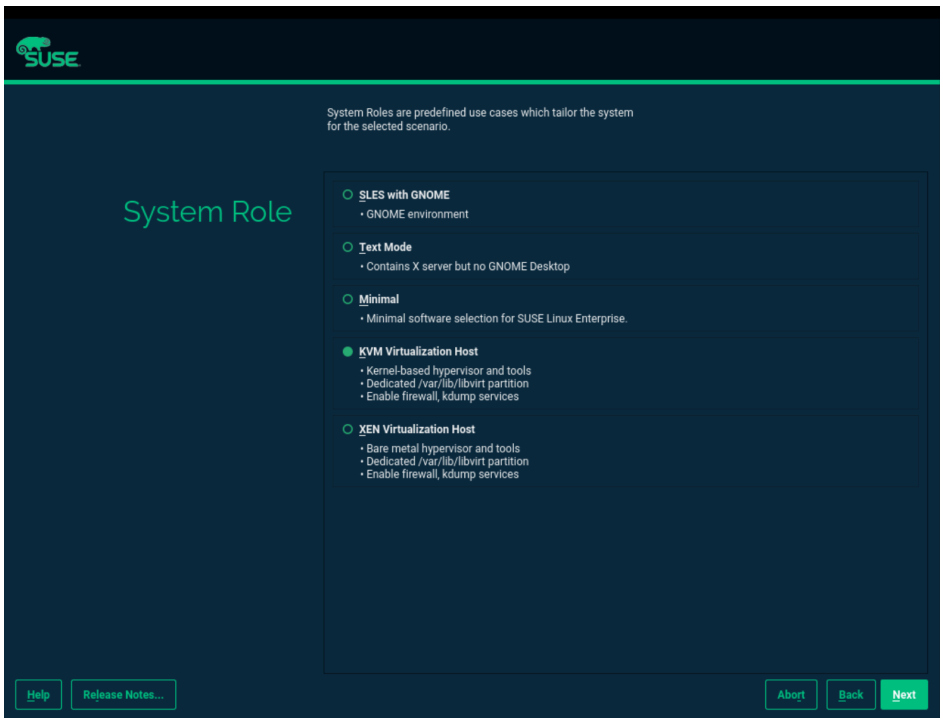


図 6.1: システムの役割を選択する画面

ここで［KVM 仮想化ホスト］もしくは［Xen 仮想化ホスト］のいずれかの役割を選択してください。これで必要なソフトウエアが選択され、openSUSE Leap のインストールと共に必要なツールもインストールされるようになります。

ヒント

どちらの仮想化ホストを選択した場合でも、`/var/lib/libvirt` にマウントされる専用のパーティションが作成されるほか、`firewalld` と Kdump サービスの有効化がそれぞれ設定されます。

6.2.2 YaST 仮想化 モジュールの実行

VM ホストサーバ 側での openSUSE Leap のインストール指定にもよりますが、通常はお使いのシステム内に仮想化ツールは全くインストールされません。必要なツールは YaST 仮想化 モジュールでハイパーバイザを選択することによって、インストールを行うことができます。



ヒント

なお、YaST 仮想化 モジュールは `yast2-vm` パッケージ内に含まれています。仮想化コンポーネントをインストールするにあたっては、あらかじめ VM ホストサーバ 内に本パッケージをインストールしておいてください。

手順 6.1: KVM 環境のインストール

KVM 仮想化環境や関連ツールをインストールするには、下記の手順を実施します:

1. YaST を起動し、[仮想化] › [ハイパーバイザとツールのインストール] を選択します。
2. [KVM サーバ] を選択すると、最小限の QEMU ツールと KVM 環境をインストールすることができます。`libvirt` ベースの管理スタックも必要とする場合は、[KVM ツール] も選択してください。選択が終わったら、[了解] を押します。
3. YaST でのインストールを行った場合は、VM ホストサーバ の設定の際、自動的にブリッジを設定するかどうかの確認を表示します。VM ゲスト に対して適切なネットワーク環境を提供するのに必要な設定であるため、通常は [はい] を押して進めてください。ネットワークが不要である場合は、[いいえ] を押してもかまいません。
4. インストールが終わったら、そのまま VM ゲスト の設定を開始することができます。VM ホストサーバ の再起動は不要です。

手順 6.2: XEN 環境のインストール

Xen 仮想化環境をインストールするには、下記の手順を実施します:

1. YaST を起動し、[仮想化] › [ハイパーバイザとツールのインストール] を選択します。
2. [Xen サーバ] を選択すると、最小限の Xen 環境をインストールすることができます。`libvirt` ベースの管理スタックも必要とする場合は、[Xen ツール] も選択してください。選択が終わったら、[了解] を押します。
3. YaST でのインストールを行った場合は、VM ホストサーバ の設定の際、自動的にブリッジを設定するかどうかの確認を表示します。VM ゲスト に対して適切なネットワーク環境を提供するのに必要な設定であるため、通常は [はい] を押して進めてください。ネットワークが不要である場合は、[いいえ] を押してもかまいません。
4. インストールが終わったらシステムを再起動します。起動時に Xen カーネル を選択してください。



ヒント: 既定の起動カーネルについて

全ての機能が期待通りに動作することを確認したら、YaST を利用して既定で起動するカーネルを変更し、Xen を有効化したカーネルを使用するようにしてください。既定のカーネルを変更する方法については、『リファレンス』、第12章「ブートローダ GRUB 2」、12.3項「YaST によるブートローダの設定」をお読みください。

6.2.3 特定のインストールパターンの選択

openSUSE Leap のソフトウエアリポジトリでは、関連するソフトウエアパッケージをまとめて インストールパターン という形で提供しています。動作中の openSUSE Leap でこれらのパターンをインストールすることで、特定の仮想化に必要なコンポーネントをまとめてインストールすることができます。パターンをインストールするには `zypper` コマンドを下記のように入力して実行します:

```
zypper install -t pattern パターン名
```

KVM 環境向けには下記のようなパターンが用意されています:

`kvm_server`
: KVM および QEMU を利用するための基本的な環境をインストールします。

`kvm_tools`
: KVM 環境で VM ゲスト を管理したり監視したりするための `libvirt` ツールをインストールします。

Xen 環境向けには下記のようなパターンが用意されています:

`xen_server`
: 基本的な Xen VM ホストサーバ をインストールします。

`xen_tools`
: Xen 環境で VM ゲスト を管理したり監視したりするための `libvirt` ツールをインストールします。

6.3 KVM での入れ子型仮想化 (nested virtualization) の有効化



重要: 技術プレビューである件について

KVM の入れ子型仮想化は現在技術プレビューの段階にあります。そのため、テスト目的にのみ使用されるべきものであり、サポート対象にも入っておりません。

入れ子型のゲストとは、KVM ゲスト内で KVM のホストを動作させて、そのホスト内でさらに KVM ゲストを動作させることを指します。入れ子型の仮想化を説明するにあたっては、下記のような仮想化レイヤを使用します:

L0
: KVM を動作させる物理マシンを意味します。

L1
: L0 内で動作する (1 段階目の) 仮想マシンを意味します。この中ではさらに KVM を動作させますが、これを ゲストハイパーバイザ と呼びます。

L2
: L1 内で動作する (2 段階目の) 仮想マシンを意味します。これを 入れ子ゲスト と呼びます。

入れ子型の仮想化を使用することで様々なメリットを受けることができます。たとえば下記のような用途が考えられます:

- クラウド環境で必要な仮想化ソリューションを動作させ、その中に必要な仮想マシンを配置することができるようになります。
- ハイパーバイザ自身のライブマイグレーションを行うことができるようになります。もちろんその中のゲスト (仮想マシン) も一括で移行することができます。



注記

入れ子型の仮想化を行った仮想マシンの場合、ライブマイグレーションには対応していません。

- ソフトウエアの開発やテストにも使用することができます。

入れ子型の仮想化を有効化するには、まず今読み込まれているモジュールをいったん解除したあと、`nested` KVM モジュールパラメータを指定して再読み込みを行う必要があります:

- Intel CPU の場合、下記のように入力して実行します:

```
> sudo modprobe -r kvm_intel && modprobe kvm_intel nested=1
```

- AMD CPU の場合は、下記のように入力して実行します:

```
> sudo modprobe -r kvm_amd && modprobe kvm_amd nested=1
```

入れ子型の仮想化を恒久的に使用したい場合は、`/etc/modprobe.d/kvm_*.conf` ファイルを作成して、その中に `nested` KVM モジュールパラメータを指定してください:

- Intel CPU の場合は、`/etc/modprobe.d/kvm_intel.conf` ファイルを作成して、下記のような行を記述します:

```
options kvm_intel nested=1
```

- AMD CPU の場合は、`/etc/modprobe.d/kvm_amd.conf` ファイルを作成して、下記のような行を記述します:

```
options kvm_amd nested=1
```

L0 ホストが入れ子型の仮想化に対応していれば、L1 ゲストに下記のいずれかの修正を加えることで、L1 内で仮想化ができるようになります:

- QEMU のコマンドラインオプションに `-cpu host` を追加します。
- QEMU のコマンドラインオプションの `-cpu` 以下に、`vmx` (Intel CPU の場合) もしくは `svm` (AMD CPU の場合) を追加します。これにより、仮想 CPU 内で仮想化ができるようになります。

6.3.1 VMware ESX のゲストハイパーバイザとしての使用

KVM ハイパーバイザ内で入れ子型の VMware ESX ハイパーバイザを動作させると、ネットワークの接続が不安定になる事象が発生します。この問題は特に、KVM ゲストとハイパーバイザの間、もしくは KVM ゲストと外部ネットワークとの間で発生します。これは KVM ゲストの設定で、下記のような既定の CPU 設定が存在する場合に生じる問題です:

```
<cpu mode='host-model' check='partial'/>
```

この問題を解決するには、CPU の設定を下記のように変更してください:

```
[...]
```

```
<cpu mode='host-passthrough' check='none'>
 <cache mode='passthrough'/>
</cpu>
[...]
```

II libvirt を利用した仮想マシンの管理

7 libvirt デーモン 26

8 VM ホストサーバ の準備 32

9 ゲストのインストール 62

10 基本的な VM ゲスト の管理 73

11 接続と認可 93

12 高度なストレージ設定 114

13 仮想マシンマネージャ を利用した仮想マシンの設定 119

14 virsh を利用した仮想マシンの設定 140

15 AMD SEV-SNP による仮想マシンのセキュリティ強化 177

16 VM ゲスト の移行 184

17 Xen から KVM への移行ガイド 193

7 libvirt デーモン

改訂履歴

2024-06-25

KVM や Xen の機能を提供する `libvirt` を使用するにあたっては、1 つまたは複数のデーモンをインストールして有効化しておく必要があります。また `libvirt` には、モノリシック型とモジュール型という 2 種類の利用形態があります。モノリシック型の場合、`libvirt` に標準で付属する `libvirtd` という単一のデーモンを使用します。ここには主要なハイパーバイザドライバのほか、ストレージやネットワーク、ノードデバイスの管理機能など、必要な全てのセカンダリドライバも用意されています。またモノリシック型の場合、`libvirtd` は外部クライアントからの機密を保持したリモートアクセス機能も提供します。もう一方のモジュール型はしばらく後に作られた仕組みで、この場合それらの機能はそれぞれ別々のデーモンとして起動します。これにより、`libvirt` のインストールをカスタマイズすることができるようになっています。既定ではモジュール型のデーモンが有効化されますが、個別のデーモンを無効化して `libvirtd` を有効化することで、従来のモノリシック型に切り替えることもできます。

モジュール型のデーモンは、最小限の `libvirt` 機能のみを使用したい場合に有用です。たとえば仮想マシンで `libvirt` 以外が提供するストレージとネットワークを利用したい場合、`libvirt-daemon-driver-storage` や `libvirt-daemon-driver-network` などのパッケージは不要になります。典型的な例が Kubernetes で、ネットワークやストレージ、cgroup やネームスペース統合などの機能は Kubernetes 側が処理します。そのためモジュール型の場合は、`virtqemud` を提供する `libvirt-daemon-driver-QEMU` パッケージのみが必須となります。モジュール型は、実際に使用するコンポーネントの種類が少ない場合に `libvirt` のインストールを最小化できる仕組みです。

7.1 モジュール型デーモンの開始と停止

モジュール型デーモンの場合、「virt ドライバ名 d」の形式でそれぞれのデーモンが提供されています。これらは `/etc/libvirt/virt`ドライバ名`d.conf` という設定ファイルで制御することができます。SUSE では `virtqemud` と `virtxend` のハイパーバイザデーモンをサポートしているほか、下記に示すセカンダリデーモンをサポートしています:

- virtnetworkd - `libvirt` の仮想ネットワーク管理 API を提供する仮想ネットワーク管理デーモンです。たとえば virtnetworkd は、仮想マシンが使用する NAT 仮想ネットワークの作成機能などを提供します。
- virtnodedevd - `libvirt` のノードデバイス管理 API を提供するホスト物理デバイス管理デーモンです。たとえば virtnodedevd は、仮想マシンが使用する PCI デバイスをホストから切り離す機能などを提供します。

- virtnwfilterd - `libvirt` のファイアウォール管理 API を提供するホストファイアウォール管理デーモンです。たとえば virtnwfilterd は、仮想マシンに対するネットワークトラフィックのフィルタ機能などを提供します。

- virtsecretd - `libvirt` の機密管理 API を提供するホスト機密管理デーモンです。たとえば virtsecretd は、LUKS ボリュームの鍵を保存する機能などを提供します。

- virtstoraged - `libvirt` のストレージ管理 API を提供するホストストレージ管理デーモンです。たとえば virtstoraged は、様々な種類のストレージプールを作成する機能のほか、作成したプール内にボリュームを作成する機能なども提供します。

- virtinterfaced - `libvirt` のネットワークインターフェイス管理 API を提供するホスト側 NIC 管理デーモンです。たとえば virtinterfaced は、ホスト側のボンディングデバイスの作成機能などを提供します。ただし SUSE では、wicked や NetworkManager などのネットワーク管理ツールを使用するのが一般的であることから、本デーモンの使用は非推奨としております。そのため virtinterfaced についても無効化しておくことをお勧めします。

- virtproxyd - 従来型の `libvirtd` ソケットとモジュール型のデーモンソケットとの仲介を行うデーモンです。`libvirt` をモジュール型で使用する場合、virtproxyd を介することで、モノリシック型の `libvirtd` に似た API を利用できるようになります。モノリシック型の `libvirtd` ソケットに接続するクライアントからも使用することができます。

- virtlogd - 仮想マシンのコンソールに出力されるログを収集するデーモンです。virtlogd はモノリシック型の `libvirtd` でも使用されているデーモンですが、virtqemud の `systemd` ユニットファイルから virtlogd を開始するように設定していることから、モジュール型の `libvirtd` では明示的に開始する必要はありません。

- virtlockd - ディスクなどの仮想マシンリソースに対して施錠 (ロック) を行うためのデーモンです。virtlockd はモノリシック型の `libvirtd` でも使用されているデーモンですが、virtqemud や virtxend の `systemd` ユニットファイルから virtlockd を開始するように設定していることから、モジュール型の `libvirtd` では明示的に開始する必要はありません。

`virtlogd` と `virtlockd` はモノリシック型の `libvirtd` でも使用されます。また、これらのデーモンはセキュリティ上の理由から、`libvirtd` とは分離されているためです。

既定では、モジュール型デーモンは `/var/run/libvirt/virtドライバ名d-sock` および `/var/run/libvirt/virtドライバ名d-sock-ro` の 2 種類の UNIX ドメインソケットで待ち受けを行います。クライアントライブラリは通常、従来型の `/var/run/libvirt/libvirtd-sock` という UNIX ソケットファイルにアクセスしますが、これらのクライアント向けに virtproxyd というデーモンが提供されています。

また、`virtqemud` と `virtxend` は、`systemd` の事前設定で有効化されているほか、`virtnetworkd`，`virtnodedevd`，`virtnwfilterd`，`virtstoraged`，`virtsecretd` の各デーモンも、対応するパッケージがインストールされていれば事前に有効化されます。これらは利便性向上のために設定されているものですが、モジュール型のデーモンはそれぞれ個別に管理することができます:

- virt`ドライバ名`d.service - `ドライバ名` に対応する仮想化デーモンを開始するためのメインのユニットファイルです。ホストの起動時に VM を開始する必要がある場合は、こちらを起動時に開始するように設定しておくことをお勧めします。
- virt`ドライバ名`d.socket - 読み書き可能な UNIX ソケットである `/var/run/libvirt/virtドライバ名d-sock` に対応するユニットファイルです。こちらは起動時に開始するよう有効化しておくことをお勧めします。
- virt`DRIVER`d-ro.socket - 読み込み専用の UNIX ソケットである `/var/run/libvirt/virtドライバ名d-sock-ro` に対応するユニットファイルです。こちらも起動時に開始するよう有効化しておくことをお勧めします。
- virt`DRIVER`d-admin.socket - 管理用の UNIX ソケットである `/var/run/libvirt/virtドライバ名d-admin-sock` に対応するユニットファイルです。こちらも起動時に開始するよう有効化しておくことをお勧めします。

`systemd` で各種のソケットを有効化すると、virt`ドライバ名`d.conf に設定されたソケット関連の設定が無視されるようになります。無視される設定と対応するユニットファイルは下記のとおりです:

- unix_sock_group - UNIX ソケットのグループ所有者を設定します。この設定は `virtドライバ名d.socket` および `virtドライバ名d-ro.socket` ユニットファイル内の `SocketGroup` パラメータで設定します。
- unix_sock_ro_perms - 読み込み専用の UNIX ソケットのアクセス許可を設定します。この設定は `virtドライバ名d-ro.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_rw_perms - 読み書き可能な UNIX ソケットのアクセス許可を設定します。この設定は `virtドライバ名d.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_admin_perms - 管理用の UNIX ソケットのアクセス許可を設定します。この設定は `virtドライバ名d-admin.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_dir - 全ての UNIX ソケットが作成されるディレクトリを指定します。この設定は `virtドライバ名d.socket`，`virtドライバ名d-ro.socket`，`virtドライバ名d-admin.socket` の各ユニットファイル内の `ListenStream` で個別に設定します。

7.2 モノリシック型デーモンの開始と停止

モノリシック型デーモンは `libvirtd` と呼ばれ、`/etc/libvirt/libvirtd.conf` ファイルで設定を行います。`libvirtd` はいくつかの `systemd` から構成されています:

- libvirtd.service - `libvirtd` を起動するためのメインとなる `systemd` ユニットファイルです。ホストの起動時に VM を開始する必要がある場合は、`libvirtd.service` を起動時に開始するように設定しておくことをお勧めします。
- libvirtd.socket - 読み書き可能な UNIX ソケット `/var/run/libvirt/libvirt-sock` に対応するユニットファイルです。こちらも起動時に開始するよう有効化しておくことをお勧めします。
- libvirtd-ro.socket - こちらは読み込み専用の UNIX ソケットである `/var/run/libvirt/libvirt-sock-ro` に対応するユニットファイルです。こちらも起動時に開始するよう有効化しておくことをお勧めします。
- libvirtd-admin.socket - 管理用の UNIX ソケットである `/var/run/libvirt/libvirt-admin-sock` に対応するユニットファイルです。こちらも起動時に開始するよう有効化しておくことをお勧めします。
- libvirtd-tcp.socket - 非 TLS リモートアクセスのための TCP ポート 16509 に対応するユニットファイルです。こちらは管理者が適切な認証機構を設定している場合を除き、起動時に開始するように設定すべきではありません。
- libvirtd-tls.socket - TLS リモートアクセスのための TCP ポート 16509 に対応するユニットファイルです。管理者が x509 証明書と適切な認証機構を設定するまでは、起動時に開始するように設定すべきではありません。

`systemd` で各種のソケットを有効化すると、`libvirtd.conf` に設定されたソケット関連の設定が無視されるようになります。無視される設定と対応するユニットファイルは下記のとおりです:

- listen_tcp - TCP 接続は `libvirtd-tcp.socket` ユニットファイルで設定します。
- listen_tls - TLS 接続は `libvirtd-tls.socket` ユニットファイルで設定します。
- tcp_port - 非 TLS 向け TCP ポートを設定します。こちらは `libvirtd-tcp.socket` ユニットファイル内の `ListenStream` パラメータで設定します。
- tls_port - TLS 向け TCP ポートを設定します。こちらは `libvirtd-tls.socket` ユニットファイル内の `ListenStream` パラメータで設定します。
- listen_addr - 待ち受ける IP アドレスを設定します。この設定は `libvirtd-tcp.socket` または `libvirtd-tls.socket` ユニットファイル内の `ListenStream` パラメータで設定します。

- unix_sock_group - UNIX ソケットのグループ所有者を設定します。この設定は `libvirtd.socket` および `libvirtd-ro.socket` ユニットファイル内の `SocketGroup` パラメータで設定します。
- unix_sock_ro_perms - 読み込み専用の UNIX ソケットのアクセス許可を設定します。この設定は `libvirtd-ro.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_rw_perms - 読み書き可能な UNIX ソケットのアクセス許可を設定します。この設定は `libvirtd.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_admin_perms - 管理用の UNIX ソケットのアクセス許可を設定します。この設定は `libvirtd-admin.socket` ユニットファイル内の `SocketMode` パラメータで設定します。
- unix_sock_dir - 全ての UNIX ソケットが作成されるディレクトリを指定します。この設定は `libvirtd.socket` , `libvirtd-ro.socket` , `libvirtd-admin.socket` の各ユニットファイル内の `ListenStream` で個別に設定します。



重要: 矛盾関係にある libvirtd と xendomains のサービスについて

`libvirtd` を開始してもエラーになってしまう場合は、まず `xendomains` サービスが開始されていないかどうかを確認してください:

```
> systemctl is-active xendomains active
```

上記のコマンドが `active` を返した場合、`libvirtd` を開始するには、事前に `xendomains` を停止させる必要があります。また、システムの起動時に `libvirtd` を開始したい場合は、これに加えて `xendomains` が自動的に開始されないように設定する必要もあります。具体的には、下記のように入力して実行してください:

```
> sudo systemctl stop xendomains
> sudo systemctl disable xendomains
> sudo systemctl start libvirtd
```

`xendomains` と `libvirtd` は同一のサービスを提供するものであり、同時に使用しようとすると、互いに競合する結果になります。たとえば `libvirtd` によって domU が起動されている場合でも、`xendomains` を開始すると、domU を再度起動しようとしてしまいます。

7.3 モノリシック型デーモンへの切り替え

モジュール型のデーモンからモノリシック型のデーモンに切り替えたい場合は、いくつかのサービスに対して変更を行う必要があります。なお、切り替えに際しては事前に動作中の仮想マシンを停止させるか、別のホストに移行しておくことをお勧めします。

1. まずはモジュール型のデーモンとソケットをそれぞれ停止します。下記の例では、KVM 向けの QEMU といくつかのセカンダリデーモンをそれぞれ停止しています。

```
for drv in qemu network nodedev nwfilter secret storage
do
 > sudo systemctl stop virt${drv}d.service
 > sudo systemctl stop virt${drv}d{,-ro,-admin}.socket
done
```

2. 次回のシステム起動時にモジュール型のデーモンが開始しないように設定します

```
for drv in qemu network nodedev nwfilter secret storage
do
 > sudo systemctl disable virt${drv}d.service
 > sudo systemctl disable virt${drv}d{,-ro,-admin}.socket
done
```

3. あとはモノリシック型の `libvirtd` のサービスとソケットを有効化します

```
> sudo systemctl enable libvirtd.service
> sudo systemctl enable libvirtd{,-ro,-admin}.socket
```

4. 最後にモノリシック型の `libvirtd` ソケットを開始します

```
> sudo systemctl start libvirtd{,-ro,-admin}.socket
```

8 VM ホストサーバ の準備

改訂履歴

2024-06-27

ゲスト側の仮想マシンをインストールする前に、まずは VM ホストサーバ 内に必要なリソースを用意して環境を整える必要があります。具体的には下記の設定を行う必要があります:

- ネットワーク: ホストが提供するネットワークと接続するための機能に関する設定を行います。
- ストレージプール: ホストが提供するストレージをゲスト側からアクセスできるようにして、ディスクイメージ等を保存できるようにします。

8.1 ネットワークの設定

VM ゲスト に対してネットワーク機能を提供するには、下記の 2 種類の方法があります:

- ネットワークブリッジ: ゲスト側にネットワーク接続を提供する設定の既定値で、推奨される方式です。
- 仮想ネットワーク: 転送機能を利用してネットワーク接続を提供する方式です。

8.1.1 ネットワークブリッジ

ネットワークブリッジは、VM ゲスト に対してレイヤ 2 スイッチの機能を提供します。レイヤ 2 スイッチは、ポート間のパケット転送の際にレイヤ 2 イーサネットパケットを利用し、MAC アドレスをベースにして宛先を判断します。これにより、VM ホストサーバ から VM ゲスト に対して、レイヤ 2 アクセスのネットワークを提供することになります。これは、VM ゲスト の仮想的なイーサネットケーブルがイーサネットハブに接続され、そこからホスト自身やホスト内で動作する他の VM ゲスト に接続できる形態と同じになります。この設定は、共有物理デバイス と称することもあります。

ネットワークブリッジは、openSUSE Leap を KVM または Xen のハイパーバイザとして設定した際の既定の設定になっています。これは VM ゲスト と VM ホストサーバ の LAN を単純に接続できるため、お勧めの方法でもあります。

ネットワークブリッジを作成する際に使用すべきツールは、VM ホストサーバ 側でのネットワーク接続を管理する際に使用しているサービスによって異なります:

- ネットワーク接続を `wicked` で管理している場合は、YaST またはコマンドラインツールを利用してネットワークブリッジを作成します。サーバ用途として構築した場合は、`wicked` が既定で使用されます。
- ネットワーク接続を NetworkManager で管理している場合は、コマンドラインツール `nmcli` でネットワークブリッジを作成します。デスクトップ環境やラップトップ環境の場合は、NetworkManager が既定で使用されます。

8.1.1.1 YaST を利用したネットワークブリッジの管理

本章では、YaST を利用してネットワークブリッジを追加または削除する方法を説明しています。

8.1.1.1.1 ネットワークブリッジの追加

VM ホストサーバ にネットワークブリッジを追加するには、下記の手順を行います:

1. [YaST] › [システム] › [ネットワークの設定] を起動します。
2. [概要] タブに移動して [追加] を押します。
3. [デバイスの種類] では [ブリッジ] を選択し、[設定名] の項目にはブリッジデバイスのインターフェイス名を入力します。あとは [次へ] を押します。
4. [アドレス] のタブでは、DHCP 経由でアドレスを取得するか、もしくは固定で IP アドレスを設定するかを選択し、必要であれば IP アドレスやサブネットマスク、ホスト名などをそれぞれ入力します。
   [可変 IP アドレス] は、ブリッジが DHCP サーバに接続されているような場合のみ有用です。物理的なイーサネットデバイスへの接続を持たない仮想ブリッジを作成する場合は、[固定 IP アドレス] を選択します。この場合、プライベート IP アドレスの範囲から選んで使用することをお勧めします。たとえば `192.168.0.0/16` , `172.16.0.0/12` , `10.0.0.0/8` などの中から選択します。
   ホストシステムとは接続しないゲスト間のみのネットワーク接続を作成する場合は、IP アドレスを `0.0.0.0` 、サブネットマスクを `255.255.255.255` に設定します。これにより、IP アドレスを設定しない特殊なネットワークを構成することができます。
5. [ブリッジ接続デバイス] のタブでは、ネットワークブリッジに含めたいネットワークデバイスを選択します。

6. ［次へ］を押すと［概要］タブに戻りますので、設定内容を確認して［OK］を押します。これで VM ホストサーバ 内にネットワークブリッジが作成され、有効化されます。

8.1.1.1.2 ネットワークブリッジの削除

既存のネットワークブリッジを削除するには、下記の手順を行います:

1. ［YaST］› ［システム］› ［ネットワークの設定］を起動します。
2. ［概要］タブで削除したいブリッジデバイスを選択します。
3. ［Delete］を押してブリッジを削除し、［OK］を押します。

8.1.1.2 コマンドラインを利用したネットワークブリッジの管理

本章では、コマンドラインを利用してネットワークブリッジを追加もしくは削除する方法を説明しています。

8.1.1.2.1 ネットワークブリッジの追加

VM ホストサーバ 内にネットワークブリッジを追加するには、下記の手順を行います:

1. ネットワークブリッジを作成したい VM ホストサーバ にログインし、`root` になります。
2. まずは新しいブリッジの名前を選択します。下記の例では、`virbr_test` という名前で作成するものとします。

```
# ip link add name virbr_test type bridge
```

3. VM ホストサーバ でブリッジが作成されたことを確認します:

```
# bridge vlan
[...]
virbr_test  1 PVID Egress Untagged
```

`virbr_test` という名前が現れていますが、この時点ではどの物理ネットワークインターフェイスにも結びつけられていません。

4. ネットワークブリッジを起動して、ブリッジにネットワークインターフェイスを追加します:

```
# ip link set virbr_test up
# ip link set eth1 master virbr_test
```

重要: ネットワークインターフェイスを使用していてはならない件について

ここで指定できるネットワークインターフェイスは、他のネットワークブリッジで使用されているものであってはなりません。

5. なお、必要であれば STP (スパニングツリープロトコル (https://ja.wikipedia.org/wiki/%E3%82%B9%E3%83%91%E3%83%8B%E3%83%B3%E3%82%B0%E3%83%84%E3%83%AA%E3%83%BC%E3%83%97%E3%83%AD%E3%83%88%E3%82%B3%E3%83%AB)) を有効化することもできます:

```
# bridge link set dev virbr_test cost 4
```

8.1.1.2.2 ネットワークブリッジの削除

コマンドラインを利用して VM ホストサーバ のネットワークブリッジを削除するには、下記の手順を行います:

1. ネットワークブリッジを削除したい VM ホストサーバ にログインし、`root` になります。
2. まずは既存のネットワークブリッジの一覧を表示させて、名前を確認します:

```
# bridge vlan
[...]
virbr_test  1 PVID Egress Untagged
```

3. ブリッジを削除します:

```
# ip link delete dev virbr_test
```

8.1.1.3 `nmcli` を利用したネットワークブリッジの作成

本章では、NetworkManager が提供するコマンドラインツール `nmcli` を利用して、ネットワークブリッジを追加する方法を説明しています。

1. まずは既存のネットワーク接続を確認します:

```
> sudo nmcli connection show --active
NAME                    UUID                                  TYPE      DEVICE
```

```
Ethernet connection 1  84ba4c22-0cfe-46b6-87bb-909be6cb1214  ethernet  eth0
```

2. `br0` という名前の新しいブリッジデバイスを作成し、確認します:

```
> sudo nmcli connection add type bridge ifname br0
Connection 'bridge-br0' (36e11b95-8d5d-4a8f-9ca3-ff4180eb89f7) \
successfully added.
> sudo nmcli connection show --active
NAME                  UUID                                  TYPE      DEVICE
bridge-br0            36e11b95-8d5d-4a8f-9ca3-ff4180eb89f7  bridge    br0
Ethernet connection 1  84ba4c22-0cfe-46b6-87bb-909be6cb1214  ethernet  eth0
```

3. 必要であれば、ブリッジの設定を表示させてもよいでしょう:

```
> sudo nmcli -f bridge connection show bridge-br0
bridge.mac-address:                     --
bridge.stp:                             yes
bridge.priority:                        32768
bridge.forward-delay:                   15
bridge.hello-time:                      2
bridge.max-age:                         20
bridge.ageing-time:                     300
bridge.group-forward-mask:              0
bridge.multicast-snooping:              yes
bridge.vlan-filtering:                  no
bridge.vlan-default-pvid:               1
bridge.vlans:                           --
```

4. 物理イーサネットデバイス `eth0` をブリッジデバイスに繋げます:

```
> sudo nmcli connection add type bridge-slave ifname eth0 master br0
```

5. あとは `eth0` インターフェイスを無効化し、新しいブリッジを有効化します:

```
> sudo nmcli connection down "Ethernet connection 1"
> sudo nmcli connection up bridge-br0
Connection successfully activated (master waiting for slaves) \
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/9)
```

8.1.1.4 VLAN インターフェイスの使用

状況によっては、2 台の VM ホストサーバ 間や VM ゲスト 間で、プライベートな接続が必要となる場合があります。たとえば VM ゲスト を異なるネットワークセグメント内にあるホストに移行させる必要がある場合や、特定の VM ゲスト のみが接続できるプライベートブリッジが必要となる場合 (異なる VM ホストサーバ 内にあってもかまいません) などがそれにあたります。このような状況下では、VLAN ネットワークを作成するのが最も簡単です。

VLAN インターフェイスは一般に VM ホストサーバ 内で設定を行います。この VLAN インターフェイスは異なる VM ホストサーバ システム間の相互接続として使用できるほか、VM ゲスト のみが接続されたブリッジへの物理インターフェイスとしても設定することができます。このほか、VM ホストサーバ では IP アドレスを設定せずに VLAN を物理インターフェイスに接続することもできてしまいます。これにより、VM ゲスト から VM ホストサーバ に接続できないように設定できることになります。

まずは YaST を起動して［システム］›［ネットワークの設定］を選択します。あとは下記の手順で VLAN インターフェイスを設定します:

手順 8.1: YAST を利用した VLAN インターフェイスの設定

1. ［追加］を押して新しいネットワークインターフェイスを作成します。
2. ［インターフェイスの追加設定］内の［デバイスの種類］では、［VLAN］を選択します。
3. ［VLAN ID］では VLAN の ID を指定します。なお、VLAN ID `1` は通常、管理用に使用します。
4. ［次へ］を押します。
5. ［VLAN の実インターフェイス］では、VLAN デバイスの接続先となるインターフェイスを選択します。一覧に必要なインターフェイスが表示されていない場合は、いったんキャンセルしてから、表示させたいインターフェイスを設定してください (このとき、IP アドレスは指定しなくてかまいません) 。
6. さらに VLAN デバイスへの IP アドレス設定方法を選択します。
7. ［次へ］を押して設定を完了してください。

VLAN インターフェイスをブリッジの物理インターフェイスとして使用することもできます。これにより、特定の VM ホストサーバ と VM ゲスト のみを接続することができるようになりますので、そのネットワークを介して VM ゲスト を移行できるようになります。

場合によっては、YaST で IP アドレスの設定を無くすことができない場合があります。たとえば VM ゲスト のみを接続するようなネットワークがそれにあたります。このような場合は、IP アドレスに `0.0.0.0` を、サブネットマスクに `255.255.255.255` をそれぞれ指定してください。これにより IP アドレス無しでの設定ができるようになります。

8.1.2 仮想ネットワーク

`libvirt` が管理する仮想ネットワークはブリッジ型のネットワークと似ていますが、一般的には VM ホストサーバ とのレイヤ 2 接続は行いません。VM ホストサーバ の物理ネットワークへの接続は、レイヤ 3 転送を利用して実現します。これは VM ホストサーバ 側でパケット転送の機能を提供するもの

で、レイヤ 2 ブリッジ型ネットワークとは異なる方式になります。この種類の仮想ネットワークでは VM ゲスト に対して DHCP や DNS のサービスを提供することもできます。`libvirt` の仮想ネットワーク機能に関する詳細は、https://libvirt.org/formatnetwork.html にある Network XML format の章をお読みください。

openSUSE Leap で標準的な方法で `libvirt` のインストールを行うと、`default` という名前の仮想ネットワークが作成され、DHCP と DNS の機能がそれぞれ提供されるようになります。また、この仮想ネットワークにはアドレス変換 (NAT) 機能が提供され、VM ホストサーバ の物理ネットワークに接続できるようになります。これはあらかじめ設定済みの形で提供されますが、管理者側で有効化する必要があります。`libvirt` でサポートされている転送モードの詳細について、詳しくは https://libvirt.org/formatnetwork.html#elementsConnect にある Network XML format ドキュメンテーションをお読みください。

`libvirt` が管理する仮想ネットワークは幅広い用途に対応していて、必要な機能のほとんどに対応していますが、ラップトップなどの無線接続や動的な (間欠的な) 接続の場合には不十分です。また仮想ネットワークは、仮想ネットワークと VM ホストサーバ のネットワークとの間でパケット転送を行うことから、VM ホストサーバ の接続しているネットワーク内で IP アドレス数が不足しているような環境に有用でもあります。しかしながら、サーバ用途の場合は、VM ゲスト を VM ホストサーバ の LAN に接続するネットワークブリッジ型の構成のほうが便利ではあります。



警告: 転送モードの有効化について

`libvirt` の仮想ネットワークに対して転送モードを有効化するには、まず `/proc/sys/net/ipv4/ip_forward` と `/proc/sys/net/ipv6/conf/all/forwarding` をそれぞれ 1 に設定する必要があります。これにより、VM ホストサーバ をルータとして動作させることができるようになります。なお、VM ホストサーバ のネットワークを再起動してしまうと、これらの値はリセットされ転送機能が無効化されてしまいます。再起動後も転送モードを有効化したい場合は、VM ホストサーバ 内の `/etc/sysctl.conf` ファイルを編集して、下記の内容を追記してください:

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

```
net.ipv6.conf.all.forwarding = 1
```

8.1.2.1 仮想マシンマネージャ を利用した仮想ネットワークの管理

仮想マシンマネージャ を利用することで、仮想ネットワークの作成や設定、操作などを行うことができます。

8.1.2.1.1 仮想ネットワークの作成

1. まずは 仮想マシンマネージャ を起動します。利用可能な接続が表示されますので、設定したい仮想ネットワークの名前を選択して［詳細］を押します。

2. ［接続の詳細］ウインドウで［仮想ネットワーク］タブを選択します。ここには現在の接続で利用可能な全ての仮想ネットワークが表示されるほか、右側には選択した仮想ネットワークの詳細が表示されます。

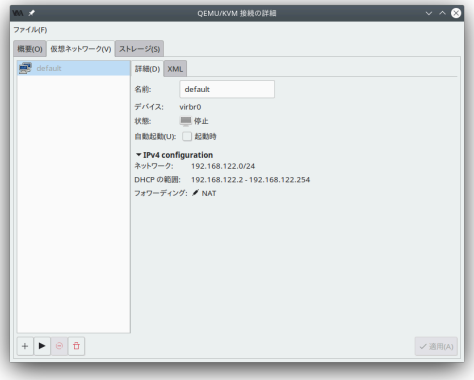


図 8.1: 接続の詳細

3. 新しい仮想ネットワークを追加するには、［追加］を押します。

4. まずは新しい仮想ネットワークの名前を指定します。

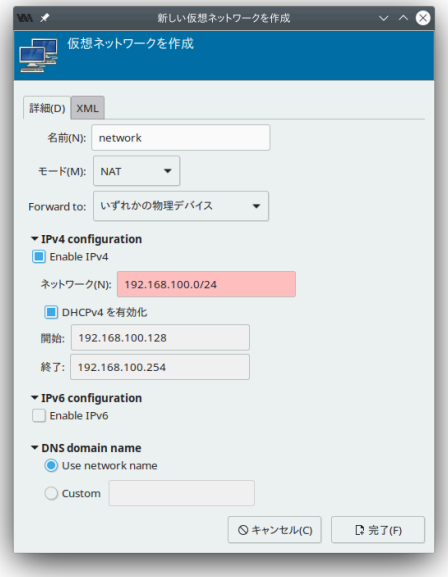


図 8.2: 仮想ネットワークの作成

5. 次にネットワークモードを指定します。[NAT] および [ルーティング] の場合は、ネットワーク通信の転送先デバイスを指定することができます。[NAT] (Network Address Translation; ネットワークアドレス変換) は仮想的なネットワークアドレス領域を特定の IP アドレスに割り当てるための方式で、1 つの IP アドレスを複数のゲストで共有することができるようになります。[ルーティング] は仮想ネットワーク側からのパケットを、そのまま VM ホストサーバ の物理ネットワークに転送する方式です。

6. IPv4 ネットワークを使用する場合は [Enable IPv4] を選択して利用したい IPv4 ネットワークアドレスを入力します。DHCP サーバが必要な場合は、[Enable DHCPv4] を選択して、割り当てるべきアドレス範囲を指定します。

7. IPv6 ネットワークを使用する場合は [Enable IPv6] を選択して利用したい IPv6 ネットワークアドレスを入力します。DHCP サーバが必要な場合は、[Enable DHCPv6] を選択して、割り当てるべきアドレス範囲を指定します。

8. 仮想ネットワークとは異なるドメイン名を設定したい場合は、[DNS domain name] で [Custom] を選択し、ドメイン名を入力します。

9. [完了] を押すと新しい仮想ネットワークを作成することができます。VM ホストサーバ 側では新しい仮想ネットワークブリッジである `virbrX` が作成され、これが新しい仮想ネットワークになります。これは `bridge link` コマンドで確認することができます。`libvirt` では自動的に iptables のルールを追加して、新しく作成した virbr`X` でのゲスト間の通信可否を設定します。

8.1.2.1.2 仮想ネットワークの起動

停止済みの仮想ネットワークを起動するには、下記の手順で行います:

1. まずは 仮想マシンマネージャ を起動します。利用可能な接続が表示されますので、設定したい仮想ネットワークの名前を選択して［詳細］を押します。
2. ［接続の詳細］ウインドウ内では［仮想ネットワーク］タブを選択します。ここには現在の接続で利用可能な、全ての仮想ネットワークが一覧表示されます。
3. 仮想ネットワークを起動するには、［開始］を押します。

8.1.2.1.3 仮想ネットワークの停止

起動済みの仮想ネットワークを停止するには、下記の手順で行います:

1. まずは 仮想マシンマネージャ を起動します。利用可能な接続が表示されますので、設定したい仮想ネットワークの名前を選択して［詳細］を押します。
2. ［接続の詳細］ウインドウ内では［仮想ネットワーク］タブを選択します。ここには現在の接続で利用可能な、全ての仮想ネットワークが一覧表示されます。
3. 停止させたい仮想ネットワークを選択して、［停止］を押します。

8.1.2.1.4 仮想ネットワークの削除

VM ホストサーバ から仮想ネットワークを削除するには、下記の手順で行います:

1. まずは 仮想マシンマネージャ を起動します。利用可能な接続が表示されますので、設定したい仮想ネットワークの名前を選択して［詳細］を押します。
2. ［接続の詳細］ウインドウ内では［仮想ネットワーク］タブを選択します。ここには現在の接続で利用可能な、全ての仮想ネットワークが一覧表示されます。
3. 削除したい仮想ネットワークを選択して［削除］を押します。

8.1.2.1.5 NAT ネットワークでの `nsswitch` を利用した IP アドレスの取得 (KVM)

- VM ホストサーバ 側に libvirt-nss をインストールします。これは libvirt 向けに NSS サポートを提供するものです:

```
> sudo zypper in libvirt-nss
```

- `/etc/nsswitch.conf` に `libvirt` を追記します:

```
...
hosts:  files libvirt mdns_minimal [NOTFOUND=return] dns
...
```

- NSCD を起動している場合は、再起動します:

```
> sudo systemctl restart nscd
```

これでホスト側からゲストを名前で接続できるようになります。

なお、NSS モジュールの機能は完全ではありません。`dnsmasq` が提供する `/var/lib/libvirt/dnsmasq/*.status` ファイルを読み込んで、JSON レコード形式で記述されているホスト名と IP アドレスを検出します。ホスト名の変換は、`dnsmasq` の動作している libvirt 管理下のブリッジ型ネットワークの VM ホストサーバ でのみ動作します。

8.1.2.2 `virsh` を利用した仮想ネットワークの管理

`libvirt` が提供する仮想ネットワークは、`virsh` コマンドラインツールで管理することができます。`virsh` で利用可能なネットワーク関連のコマンドを一覧表示するには、下記のように入力して実行します:

```
> sudo virsh help network
Networking (help keyword 'network'):
    net-autostart                  ネットワークの自動起動
    net-create                     XML ファイルによるネットワークの作成
    net-define                     define (but don't start) a network from an XML file
    net-destroy                    ネットワークの強制停止
    net-dumpxml                    XML 形式のネットワーク情報
    net-edit                       ネットワークの XML 設定の編集
    net-event                      Network Events
    net-info                       ネットワーク情報
    net-list                       ネットワークの一覧表示
    net-name                       ネットワーク UUID からネットワーク名への変換
    net-start                      停止状態の ( 定義済み ) ネットワークの起動
    net-undefine                   undefine an inactive network
    net-update                     既存のネットワーク設定の一部分の更新
```

```
    net-uuid                       ネットワーク名からネットワーク UUID への変換
    net-port-list                  list network ports
    net-port-create                create a network port from an XML file
    net-port-dumpxml               network port information in XML
    net-port-delete                delete the specified network port
```

`virsh` の特定のコマンドに対するヘルプを表示したい場合は、`virsh help コマンド` のように入力して実行してください。

```
> sudo virsh help net-create
  名前
    net-create - XML ファイルによるネットワークの作成

  形式
    net-create <file>

  詳細
    ネットワークを作成します。

  オプション
    [--file] <string>  ネットワーク の XML 記述を含むファイル
```

8.1.2.2.1 ネットワークの作成

新しい 動作中 の仮想ネットワークを作成するには、下記のように入力して実行します:

```
> sudo virsh net-create 定義ファイル名.xml
```

ここで、`定義ファイル名.xml` には、`libvirt` が受け入れ可能な XML 形式で記述された、仮想ネットワークの設定ファイルを指定します。

有効化せずに新しい仮想ネットワークを追加したい場合は、下記のように入力して実行します:

```
> sudo virsh net-define 定義ファイル名.xml
```

下記には、様々な種類の仮想ネットワークの設定例を示しています。

例 8.1: NAT ベースのネットワーク

> 下記の設定は VM ゲスト から外部に発信する通信を許可するネットワーク設定です。ただし、VM ホストサーバ 側で外部に接続することのできる環境が必要です。VM ホストサーバ 側にそのようなネットワークが用意されていない場合、相互に通信できるだけのネットワークになります。
>
> ```
> <network>
> <name>vnet_nated</name> ❶
> <bridge name="virbr1"/> ❷
>  <forward mode="nat"/> ❸
>  <ip address="192.168.122.1" netmask="255.255.255.0"> ❹
> ```

```
  <dhcp>
   <range start="192.168.122.2" end="192.168.122.254"/> ❺
   <host mac="52:54:00:c7:92:da" name="host1.testing.com" \
    ip="192.168.1.101"/> ❻
   <host mac="52:54:00:c7:92:db" name="host2.testing.com" \
    ip="192.168.1.102"/>
   <host mac="52:54:00:c7:92:dc" name="host3.testing.com" \
    ip="192.168.1.103"/>
  </dhcp>
 </ip>
</network>
```

❶ 新しい仮想ネットワークの名前です。

❷ 仮想ネットワークを構築する際のブリッジデバイス名を指定します。<forward> (転送) モードが `"nat"` もしくは `"route"` のネットワークを作成する場合 (もしくは <forward> 要素を指定せず、孤立したネットワークを作成する場合) で、何も名前を指定しない場合、`libvirt` が自動的にユニークな名前を生成して使用します。

❸ <forward> 要素を含めることで、仮想ネットワークを物理 LAN に接続することができるようになります。`mode` (モード) 属性は転送方法を指定する属性で、最もよく使用されるものが "nat" (Network Address Translation (ネットワークアドレス変換); 既定値), "route" (アドレス変換を行わずに直接物理ネットワークに転送), "bridge" (`libvirt` の外部で設定されるブリッジに接続) のいずれかです。<forward> 要素を何も指定しないと、その仮想ネットワークは他のネットワークから隔離 (孤立) されることになります。それ以外の転送モードについて、詳しくは https://libvirt.org/formatnetwork.html#elementsConnect をお読みください。

❹ ネットワークブリッジに設定する IP アドレスとネットマスクです。

❺ 仮想ネットワークに対して DHCP サーバ機能を有効化し、`start` 属性と `end` 属性で指定した範囲の IP アドレスを提供します。

❻ 指定は任意ですが、<host> 要素を設定することで、内蔵の DHCP サーバに対して割り当てるべき IP アドレスを設定することもできます。この <host> 要素では、IPv4 の場合は割当先のホストの MAC アドレスと DHCP サーバが割り当てるべき IP アドレス、そしてホスト名をそれぞれ設定します。IPv6 の場合は IPv4 と少し異なり、IPv6 では使用しない `mac` 属性が存在しない代わりに、`name` 属性でホストを識別します。DHCPv6 の場合、`name` 属性は対象の純粋なホスト名である必要があります。なお、IPv4 でも `mac` の代わりにホスト名を使用することができます。

例 8.2: ルーティング型のネットワーク

下記の設定は、仮想ネットワーク内のトラフィックを LAN にそのまま (NAT を適用せずに) 転送する場合の例を示しています。IP アドレスの範囲は、VM ホストサーバ 側のネットワークルータであらかじめ設定されていなければなりません。

```
<network>
 <name>vnet_routed</name>
 <bridge name="virbr1"/>
 <forward mode="route" dev="eth1"/>(1)
 <ip address="192.168.122.1" netmask="255.255.255.0">
  <dhcp>
   <range start="192.168.122.2" end="192.168.122.254"/>
  </dhcp>
 </ip>
</network>
```

(1) この例では、ゲスト側のトラフィックは VM ホストサーバ 内の `eth1` デバイスを介して送信されます。

例 8.3: 孤立したネットワーク

この設定は、プライベートネットワークとして完全に孤立させる設定例となります。ゲスト同士や VM ホストサーバ との間では通信ができるものの、LAN 内のマシンからは全く接続できなくなります。これは XML の設定内に <forward> 要素が存在していないためです。

```
<network>
 <name>vnet_isolated</name>
 <bridge name="virbr3"/>
 <ip address="192.168.152.1" netmask="255.255.255.0">
  <dhcp>
   <range start="192.168.152.2" end="192.168.152.254"/>
  </dhcp>
 </ip>
 </network>
```

例 8.4: VM ホストサーバ にある既存のブリッジの使用

この設定は、VM ホストサーバ 内にある既存のネットワークブリッジ `br0` を使用する場合の例です。VM ゲスト は物理ネットワークに直接接続される形になります。VM ゲスト の IP アドレスは物理ネットワークと同じサブネットでなければなりませんが、その代わり、接続に関する制限は送受信ともになくなります。

```
<network>
        <name>host-bridge</name>
        <forward mode="bridge"/>
        <bridge name="br0"/>
</network>
```

8.1.2.2.2 ネットワークの一覧表示

`libvirt` で利用可能な仮想ネットワークの一覧を表示するには、下記のように入力して実行します:

```
> sudo virsh net-list --all
```

```
 名前                  状態        自動起動       永続
----------------------------------------------------------
 crowbar               動作中      はい (yes)     はい (yes)
 vnet_nated            動作中      はい (yes)     はい (yes)
 vnet_routed           動作中      はい (yes)     はい (yes)
 vnet_isolated         停止状態    はい (yes)     はい (yes)
```

利用可能なドメインの一覧を表示するには、下記のように入力して実行します:

```
> sudo virsh list
 Id    名前                           状態
----------------------------------------------------
 1     nated_sles12sp3                動作中
 ...
```

動作中のドメインのインターフェイス一覧を表示したい場合は、`domifaddr ドメイン` のように入力して実行します。このとき、インターフェイス名を直接指定して実行することもできます。既定では IP アドレスと MAC アドレスがそれぞれ出力されます:

```
> sudo virsh domifaddr nated_sles12sp3 --interface vnet0 --source lease
 名前       MAC アドレス          Protocol     Address
-------------------------------------------------------------------------------
 vnet0      52:54:00:9e:0d:2b    ipv6         fd00:dead:beef:55::140/64
 -          -                    ipv4         192.168.100.168/24
```

指定したドメインに関連づけられた全ての仮想インターフェイスの概要情報を表示するには、下記のように入力して実行します:

```
> sudo virsh domiflist nated_sles12sp3
Interface  Type       Source       Model       MAC
-------------------------------------------------------
vnet0      network    vnet_nated   virtio      52:54:00:9e:0d:2b
```

8.1.2.2.3 ネットワークの関する詳細の取得

ネットワークに関する詳細情報を取得するには、下記のように入力して実行します:

```
> sudo virsh net-info vnet_routed
名前:           vnet_routed
UUID:           756b48ff-d0c6-4c0a-804c-86c4c832a498
起動中:         はい (yes)
永続:           はい (yes)
自動起動:       はい (yes)
ブリッジ:       virbr5
```

8.1.2.2.4 ネットワークの起動

設定済みではあるものの、現在起動していないネットワークを起動したい場合は、まず対象の名前 (もしくは識別子や UUID) を判断します。具体的には、下記のように入力して実行します:

```
> sudo virsh net-list --inactive
 名前                 状態         自動起動       永続
----------------------------------------------------------
 vnet_isolated        停止状態     はい (yes)     はい (yes)
```

あとは下記のように入力して実行します:

```
> sudo virsh net-start vnet_isolated
ネットワーク vnet_isolated を起動しました
```

8.1.2.2.5 ネットワークの停止

動作中のネットワークを停止するには、まず対象の名前 (もしくは識別子や UUID) を判断します。具体的には、下記のように入力して実行します:

```
> sudo virsh net-list --inactive
 名前                 状態         自動起動       永続
----------------------------------------------------------
 vnet_isolated        動作中       はい (yes)     はい (yes)
```

あとは下記のように入力して実行します:

```
> sudo virsh net-destroy vnet_isolated
ネットワーク vnet_isolated は強制停止されました
```

8.1.2.2.6 ネットワークの削除

停止したネットワークの設定を VM ホストサーバ から完全に停止したい場合は、下記のように入力して実行します:

```
> sudo virsh net-undefine vnet_isolated
ネットワーク vnet_isolated の定義が削除されました
```

8.2 ストレージプールの設定

VM ホストサーバ 自身から VM ゲスト を管理する場合であれば、VM ホストサーバ のファイルシステム内で任意の箇所にアクセスして仮想ハードディスクを割り当てたり作成したりすることができますし、既存のイメージを VM ゲスト に割り当てたりすることができます。しかしながら、このような機

能は VM ゲスト をリモートから管理しようとしている場合には実現できません。このような理由から、`libvirt` では「ストレージプール」と呼ばれる機能を提供して、リモートからアクセスできるようにしています。



ヒント: CD/DVD ISO イメージ

リモートのクライアントから VM ホストサーバ 内の CD/DVD イメージにアクセスできるようにするには、まずそれらのイメージをストレージプール内に配置します。

`libvirt` では 2 種類のストレージを管理しています。それらはボリュームとプールと呼ばれます。

ストレージボリューム

ストレージボリュームは、ゲストに対して割り当てることのできるストレージデバイスを意味します。たとえば仮想ディスクや CD/DVD イメージのほか、フロッピィディスクイメージなども該当します。また、ストレージボリュームを物理デバイス (パーティションや論理ボリュームなど) に設定することもできますし、VM ホストサーバ 内のファイルとして存在させることもできます。

ストレージプール

ストレージプールは VM ホストサーバ 内のストレージリソースを意味するもので、その中にストレージボリュームが存在する形になります。デスクトップマシンにおけるネットワークストレージのような存在です。物理的には下記のいずれかの形態を取ることができます:

ファイルシステム内のディレクトリ ([dir])

イメージファイルを含むディレクトリの形態です。イメージファイルにはディスク形式 (raw, qcow2) のほか、ISO イメージであってもかまいません。

物理ディスクデバイス ([disk])

物理ディスク全体をストレージプールとして使用する形態です。その中にパーティションを作成してストレージボリュームを設定し、プールへの追加を行います。

フォーマット済みブロックデバイス ([fs])

ファイルシステムのディレクトリをストレージプールとして使用する場合と同様に、特定のパーティションをストレージプールとして使用する形態です。ただし、この方式の場合は、`libvirt` 側でデバイスのマウント処理を行います。

iSCSI ターゲット (iscsi)

iSCSI ターゲット内にストレージプールを作成する形態です。この場合は、`libvirt` で使用するより前に、あらかじめボリュームにログインしておく必要があります。YaST の [iSCSI イニシエータ] モジュールを利用してターゲットを検出し、ログインしておいてください。iSCSI ストレージプール内でのボリューム作成はサポートされていませんが、既存の論理ユニット番号 (LUN) そのものがボリュームそのものになります。それぞれのボリュー

ム (LUN) を使用するには、あらかじめパーティションテーブルを作成しておくか、ディスクラベルを設定しておく必要があります。いずれも存在していない場合は、`fdisk` で設定してください:

```
> sudo fdisk -cu /dev/disk/by-path/ip-192.168.2.100:3260-iscsi-
iqn.2010-10.com.example:[...]-lun-2
デバイスには認識可能なパーティション情報が含まれていません。
新しい DOS ディスクラベルを作成しました。識別子は 0xdeadbeef です。

コマンド (m でヘルプ): w
パーティション情報が変更されました。
ディスクを同期しています。
```

LVM ボリュームグループ (logical)

LVM のボリュームグループをプールとして使用する形態です。設定済みのボリュームグループの中から選択するか、もしくは使用したいデバイスを指定してグループを作成することができます。ストレージボリュームは、その中のパーティションとして作成されます。



警告: LVM ベースのプールの削除

LVM ベースのプールをストレージマネージャで削除した場合、ボリュームグループについても削除が行われます。そのため、プール内にあるデータは完全に削除され、復元できなくなることに注意してください!

マルチパスデバイス ([mpath])

現時点では、マルチパスデバイスへのサポートは限定的で、ゲストに対して既存のデバイスを割り当てることだけができます。`libvirt` 内からマルチパスデバイスのボリュームを作成する機能は提供されていません。

ネットワーク公開ディレクトリ ([netfs])

ファイルシステムのディレクトリプール (イメージファイルをディレクトリ内に配置する) と同様に、ネットワークディレクトリを指定して、そのディレクトリ内に配置する方法です。ただし、こちらの場合は `libvirt` 側でマウント処理を行います。こちらは NFS に対応しています。

SCSI ホストアダプタ ([scsi])

iSCSI ターゲットと同様に SCSI ホストアダプタを使用する方法です。この場合、デバイス名に `/dev/sdX` ではなく、`/dev/disk/by-*` を使用することをお勧めします。これは、`/dev/sdX` を使用してしまうと、ハードディスクの取り付けや取り外しなどで名前が変化してしまうことがあり得るためです。iSCSI プール内のボリューム作成には対応していませんが、既存の LUN (論理ユニット番号; Logical Unit Number) がそのままボリュームを表します。



警告: セキュリティ面の考慮事項について

データの損失や破壊を防ぐため、LVM ボリュームグループや iSCSI ターゲットなど、VM ホストサーバ のストレージプールを構成するリソースを別目的で使用してはなりません。これらに対する必要な処理は `libvirt` が行いますので、VM ホストサーバ から直接接続する必要はありませんし、VM ホストサーバ からマウントする必要もありません。

また VM ホストサーバ では、ラベルによるマウントも行わないようにしてください。ラベルによるマウントを行ってしまうと、本来は VM ゲスト 側で使用すべきパーティションを VM ホストサーバ 側が検知して、マウントしてしまう危険性があるためです。

8.2.1 `virsh` を利用したストレージの管理

コマンドラインからのストレージの管理についても、`virsh` から行うことができます。ただし、ストレージプールの作成は現時点では SUSE のサポート対象外となっております。そのため、本章ではプールの起動／停止／削除のほか、ボリュームの管理に絞って説明しています。

`virsh` で利用可能なプール／ボリューム管理関連のサブコマンド一覧を表示するには、`virsh help pool` および `virsh help volume` を実行します。

8.2.1.1 プールとボリュームの一覧表示

現時点で有効化されているプールの一覧を表示するには、下記のコマンドを入力して実行します。無効化されているプールも表示したい場合は、`--all` オプションを追加してください:

```
> virsh pool-list --details
```

特定のプールに関する詳細を表示したい場合は、`pool-info` サブコマンドを使用します:

```
> virsh pool-info プール名
```

既定では、ボリュームはプールごとに表示されます。プール内の全てのボリュームを表示したい場合は、下記のように入力して実行します:

```
> virsh vol-list --details プール名
```

現時点では、`virsh` はゲスト側で使用されているかどうかを表示する機能は提供されていません。下記の手順を行うことで、VM ゲスト 側で使用されている全てのプール内のボリュームを表示することができます。

手順 8.2: VM ホストサーバ で使用されている全てのストレージボリュームの一覧表示

1. まずは下記のような内容で XSLT スタイルシートのファイルを作成します。たとえば ~/libvirt/guest_storage_list.xsl というファイル名で保存します:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="text()"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="disk">
    <xsl:text>  </xsl:text>
    <xsl:value-of select="(source/@file|source/@dev|source/@dir)[1]"/>
    <xsl:text>&#10;</xsl:text>
  </xsl:template>
</xsl:stylesheet>
```

2. あとは下記のようなコマンドをシェル内で実行します。なお、ゲスト側の XML 設定は全て既定の場所 (`/etc/libvirt/qemu`) に保存されているものとします。また、`xsltproc` は `libxslt` パッケージ内に含まれています。

```
SSHEET="$HOME/libvirt/guest_storage_list.xsl"
cd /etc/libvirt/qemu
for FILE in *.xml; do
  basename $FILE .xml
  xsltproc $SSHEET $FILE
done
```

8.2.1.2 プールの起動／停止／削除

`virsh` pool サブコマンドを使用することで、プールの起動と停止、削除をそれぞれ行うことができます。下記では、`プール名` の箇所をプールの名前もしくは UUID に置き換えて実行してください:

プールの停止

```
> virsh pool-destroy プール名
```



注記: プールの状態は割り当てられているボリュームとは無関係である件について

既に VM ゲスト に割り当てられているプール内のボリュームについては、そのプールの状態 ([動作中] もしくは [停止]) に関わらず、常にアクセスできる状態になります。プールの状態は、リモートから VM ゲスト のボリュームを管理する場合にのみ有効となります。

プールの削除

```
> virsh pool-delete プール名
```



警告: ストレージプールの削除

警告: ストレージプールの削除 をお読みください

プールの起動

```
> virsh pool-start プール名
```

プールの自動起動の有効化

```
> virsh pool-autostart プール名
```

自動起動が有効化されたプールは、VM ホストサーバ の起動時に自動的に起動が行われるようになります。

プールの自動起動の無効化

```
> virsh pool-autostart プール名 --disable
```

8.2.1.3 ストレージプールへのボリュームの追加

`virsh` ではストレージプールにボリュームを追加するにあたって、2 種類の方法を提供しています。`vol-create` および `vol-create-from` を利用して、XML 設定ファイルから追加する方法と、`vol-create-as` のコマンドラインにパラメータを設定して追加する方法です。前者は現時点で SUSE 側でサポート対象となっていませんので、本章では `vol-create-as` を説明します。

既存のプールにボリュームを追加するには、下記のようなコマンドを入力して実行します:

```
> virsh vol-create-as プール名❶ 名前❷ 12G --format❸ raw|qcow2❹ --allocation 4G❺
```

1. ボリュームを追加したいプールの名前を指定します
2. ボリュームの名前を指定します
3. イメージのサイズを指定します。この例では 12 ギガバイトを指定していますが、k (キロバイト), M (メガバイト), G (ギガバイト), T (テラバイト) の各接尾辞を指定することができます。
4. ボリュームの形式を指定します。SUSE では `raw` と `qcow2` をサポート対象としています。
5. 任意指定のパラメータです。既定では、`virsh` はスパースファイルという形式でイメージファイルを作成し、必要に応じて自動的にサイズが拡張されるようになりますが、ここで指定したサイズ (この例では 4 ギガバイト) を事前に割り当てておくことができます。なお、k (キロバイト), M (メガバイト), G (ギガバイト), T (テラバイト) の各接尾辞を指定することができます。
   このパラメータを指定しない場合、事前の割り当ての無いスパースファイルを作成することになります。あらかじめ全てのサイズを割り当てておきたい (非スパースファイルを作成したい) 場合は、イメージのサイズと同じ値 (この例では `12G`) を指定してください。

8.2.1.3.1 既存のボリュームの複製

プールにボリュームを追加するもう 1 つの方法として、既存のボリュームの複製という手段があります。新しいボリュームは元のボリュームと同じプール内に作成されます。

```
> virsh vol-clone 既存のボリューム名❶ 新しいボリューム名❷ --pool プール❸
```

1. 複製元となる既存のボリューム名を指定します
2. 新しいボリューム名を指定します
3. 任意指定のパラメータです。`libvirt` では既存のボリュームの配置先を自動的に判断しますが、それがうまくいかない場合に、このパラメータを指定してください。

8.2.1.4 ストレージプールからのボリュームの削除

プールからボリュームを恒久的に削除したい場合は、`vol-delete` サブコマンドを使用します:

```
> virsh vol-delete ボリューム名 --pool プール名
```

`--pool` は任意指定で、通常は `libvirt` がボリュームのプールを自動的に判断しますが、それがうまくいかない場合には、このパラメータを指定してください。



警告: ボリューム削除時のチェックにいて

ボリュームは VM ゲスト 側が使用しているかどうかにかかわらず、どのような状況下であっても削除ができてしまいます。また、削除したボリュームを復元する方法はありません。

ボリュームが VM ゲスト 側で使用されているかどうかを判断したい場合は、手順8.2「VM ホストサーバ で使用されている全てのストレージボリュームの一覧表示」の手順を行ってください。

8.2.1.5 VM ゲスト へのボリュームの割り当て

8.2.1.3項「ストレージプールへのボリュームの追加」で説明している手順でボリュームを作成したら、あとは仮想マシンへの割り当てを行ってハードディスクとして使用できるようにします:

```
> virsh attach-disk ドメイン イメージファイル 接続先ディスクデバイス
```

たとえば下記のようになります:

```
> virsh attach-disk sles12sp3 /virt/images/example_disk.qcow2 sda2
```

ディスクが接続できたかどうかを確認するには、`virsh dumpxml` コマンドの出力を確認します:

```
# virsh dumpxml sles12sp3
[...]
<disk type='file' device='disk'>
 <driver name='qemu' type='raw'/>
 <source file='/virt/images/example_disk.qcow2'/>
 <backingStore/>
 <target dev='sda2' bus='scsi'/>
 <alias name='scsi0-0-0'/>
 <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
[...]
```

8.2.1.5.1 ホットプラグ (活性接続) ／恒久的な変更

ディスクの接続は、対象のドメインが動作中であっても停止済みであっても実施することができます。また、`--live` と `--config` のオプションを使用することで、それぞれ下記のような動作を行うことができます:

`--live`

動作中のドメインに対してディスクをホットプラグします。接続の情報はドメインの設定ファイル内には保存されません。また、ドメインが動作中ではない場合、`--live` オプションを指定するとエラーになります。

`--config`

ドメインの設定ファイルを恒久的に変更します。接続されたディスクは次回のドメイン起動以降に利用できるようになります。

`--live` `--config`

ディスクをホットプラグで接続すると共に、ドメインの設定ファイルを恒久的に変更します。



ヒント: `virsh attach-device`

`virsh attach-device` コマンドは `virsh attach-disk` コマンドのより一般的な形式です。このコマンドは、ドメインに対して様々な種類のデバイスを接続することができます。

8.2.1.6 VM ゲスト からのボリュームの接続解除

ドメインに対してディスクへの接続を解除したい場合は、`virsh detach-disk` コマンドを使用します:

```
# virsh detach-disk ドメイン 接続先のディスクデバイス
```

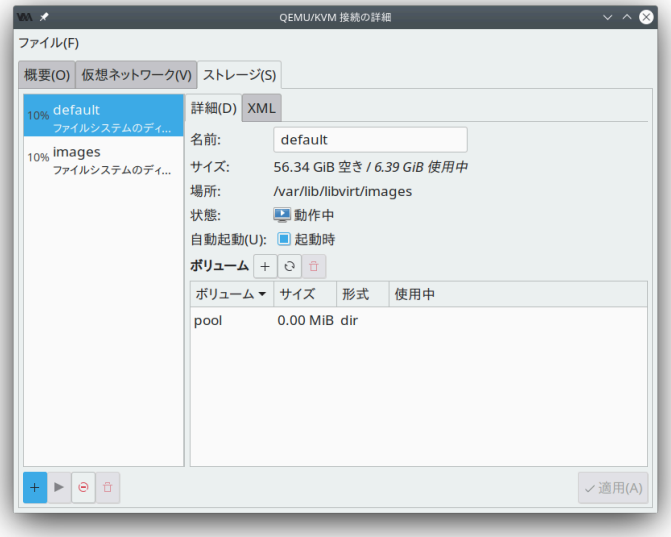
たとえば下記のようになります:

```
# virsh detach-disk sles12sp3 sda2
```

8.2.1.5項「VM ゲスト へのボリュームの割り当て」で説明しているのと同様に、こちらでも `--live` オプションや `--config` オプションで接続方式を制御することができます。

8.2.2 仮想マシンマネージャ を利用したストレージの管理

仮想マシンマネージャ はグラフィカルインターフェイスを提供するソフトウエアで、この中のストレージマネージャの機能を使用することで、ボリュームやプールを管理することができます。ストレージマネージャにアクセスするには、接続を選択してマウスの右ボタンを押し、表示されたメニューから［詳細］を選択するか、接続を選択して［編集］›［接続の詳細］を選択します。その後［ストレージ］タブを選択してください。

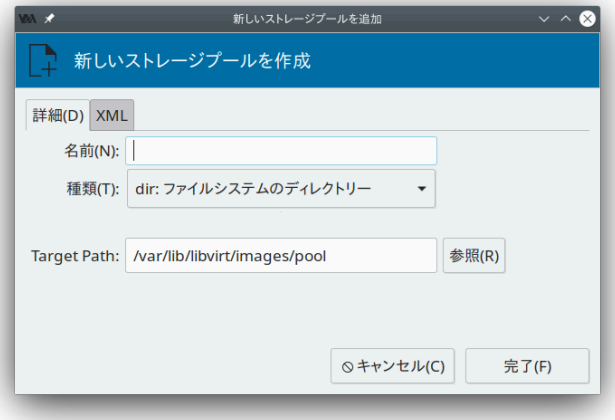


8.2.2.1 ストレージプールの追加

ストレージプールを追加するには、下記の手順を行います:

1. 左下の［追加］ボタンを押します。［新しいストレージプールを作成］というダイアログが表示されるはずです。

2. まずはプールに対する［名前］を入力します (名前には英数字のほか、_ , - , . の各記号を使用することができます) 。あとは［種類］を選択します。



3. あとはプールの種類に従って詳細を指定します。それぞれ下記のようになります:



重要

ZFS プールには対応していません。

種類 = [dir] の場合:

- [Target Path]：既存のディレクトリを選択します。

種類 = [disk] の場合:

- [フォーマット]：デバイスのパーティションテーブルの形式を指定します。通常は [auto] のままでかまいませんが、正しく検出できない場合は、VM ホストサーバ 側で `parted -l` を実行して形式を判別し、指定してください。
- [ソースパス]：デバイスのパスを指定します。`/dev/sdX` のようなパスではなく、`/dev/disk/by-*` のパスで指定することをお勧めします。これは、`/dev/sdX` を使用してしまうと、ハードディスクの取り付けや取り外しなどで名前が変化してしまうことがあり得るためです。また、パーティションではなくディスク全体を表すパスを指定してください。

種類 = [fs] の場合:

- [Target Path]：VM ホストサーバ 内のファイルシステムでのマウントポイントを指定します。
- [フォーマット]：デバイスのファイルシステム形式を指定します。既定値である `auto` で問題なく動作するはずです。
- [ソースパス]：デバイスファイルのパスを指定します。`/dev/sdX` のようなパスではなく、`/dev/disk/by-*` のパスで指定することをお勧めします。これは、`/dev/sdX` を使用してしまうと、ハードディスクの取り付けや取り外しなどで名前が変化してしまうことがあり得るためです。

種類 = [iscsi] の場合:

VM ホストサーバ 側で下記のコマンドを入力して実行し、必要なデータを収集してください:

```
> sudo iscsiadm --mode node
```

上記のように入力して実行すると、下記のような形式で iSCSI ボリュームの一覧が出力されます。下記のうち、太字の部分が必要な情報になります:

```
IP_アドレス:PORT,TPGT ターゲット名_(IQN)
```

- [Target Path]：デバイスファイルを含むディレクトリを指定します。`/dev/disk/by-path` (既定値) もしくは `/dev/disk/by-id` のいずれかを指定してください。
- [ホスト名]：iSCSI サーバのホスト名または IP アドレスを指定します。
- [ソース IQN]：iSCSI のターゲット名 (iSCSI Qualified Name (IQN)) を指定します。
- [イニシエーターの IQN]：iSCSI のイニシエータ名を指定します。

種類 = [logical] の場合

- [Volgroup Name]：既存のボリュームグループのデバイスパスを指定します。

種類 = [mpath] の場合:

- [Target Path]：マルチパス機能への対応は現在、全てのマルチパスデバイスを利用可能にする場合のみとなっています。そのため、ここで指定した任意の文字列は無視されますが、何らかの値を入力しないと XML パーサーの処理が失敗してしまいます。

種類 = [netfs] の場合:

- [Target Path]：VM ホストサーバ 内のファイルシステムでのマウントポイントを指定します。
- [ホスト名]：ネットワークファイルシステムを提供するサーバの IP アドレスまたはホスト名を指定します。
- [ソースパス]：公開されているサーバ内のディレクトリを指定します。

種類 = [rbd] の場合:

- [ホスト名]：RADOS ブロックデバイスを提供するサーバの IP アドレスまたはホスト名を指定します。
- [Source Name]：サーバ側での RADOS ブロックデバイスの名前を指定します。

種類 = [scsi] の場合

- [Target Path]：デバイスファイルを含むディレクトリを指定します。`/dev/disk/by-path` (既定値) もしくは `/dev/disk/by-id` のいずれかを指定してください。
- [ソースパス]：SCSI アダプタの名前を指定します。



注記: ファイル参照について

[参照] ボタンによるファイル参照機能は、リモートから接続している場合には使用できません。

4. [完了] を押すとストレージプールを追加することができます。

8.2.2.2 ストレージプールの管理

仮想マシンマネージャ のストレージマネージャは、プール内にボリュームを作成したり削除したりすることができます。このほか、既存のストレージプールを一時的に無効化したり恒久的に削除したりすることもできます。なお、プールの基本的な設定の変更については、SUSE でのサポート対象外となっています。

8.2.2.2.1 プールの起動／停止／削除

ストレージプールは、VM ゲスト をリモートから管理する際に、VM ホストサーバ 内に存在するブロックデバイスを VM ゲスト に追加したり削除したりする機能を提供するものです。リモートからアクセスできないように一時的に停止したい場合は、ストレージマネージャ内の左下にある [停止] ボタンを押してください。停止されたプールは [状態: 停止] となり、左側の一覧ではグレーアウト表示されます。既定では新しく作成したプールは自動的に VM ホストサーバ 側で [自動起動] が設定され、自動的に起動するようになります。

停止されているプールを起動して、リモートからアクセスできるようにしたい場合は、ストレージマネージャ内の左下にある [開始] を押します。



注記: プールの状態は割り当てられているボリュームとは無関係である件について

既に VM ゲスト に割り当てられているプール内のボリュームについては、そのプールの状態 ([動作中] もしくは [停止]) に関わらず、常にアクセスできる状態になります。プールの状態は、リモートから VM ゲスト のボリュームを管理する場合にのみ有効となります。

プールに対して恒久的にアクセスを停止したい場合は、ストレージマネージャ内の左下にある［削除］ボタンを押します。ただし、削除は停止されたプールにのみ実施することができます。なお、プールを削除しても、VM ホストサーバ 内の内容が物理的に削除されることはなく、プールの設定のみが削除されます。ただし、LVM ボリュームグループの場合は例外で、この場合は物理的にも削除されることに注意してください。



警告: ストレージプールの削除

まだ VM ゲスト に割り当てられているボリュームが存在するプールの場合、ローカルの ファイルシステムディレクトリやパーティション／ディスクをベースにしたストレージプールを削除しても、ボリュームには影響しません。

iSCSI, SCSI, LVM ボリュームグループ, ネットワーク公開ディレクトリの場合は、プールを削除すると VM ゲスト からもアクセスができなくなります。ボリュームそれ自身は削除されませんが、VM ホストサーバ からリソースへのアクセスはできなくなります。

iSCSI/SCSI ターゲットやネットワーク公開ディレクトリの場合、新しいプールを作成し直して同じ設定を作成するか、もしくはホストシステム側から対象のリソースを直接マウントすることで、再度アクセスできるようになります。

LVM ボリュームグループベースのストレージプールの場合、LVM のボリュームグループ設定が削除されることになりますので、ホストシステム内でも LVM ボリュームグループがアクセスできなくなります。この場合は設定の復元は不可能で、プール内のボリュームも恒久的に失われることになります。

8.2.2.2.2 ストレージプールへのボリュームの追加

仮想マシンマネージャ ではプールの種類がマルチパス, iSCSI, SCSI である場合を除いて、全てのストレージプール内にボリュームを作成することができます。これらのプール内のボリュームは LUN と等価な存在であり、`libvirt` 側では変更ができません。

1. 新しいボリュームはストレージマネージャを利用して作成することができるほか、VM ゲスト に新しいストレージデバイスを追加する際に作成することもできます。いずれの場合であっても、左側のパネルでストレージプールを選択して［新しいボリュームの作成］を押します。

2. ［名前］欄にイメージの名前を入力して、イメージの形式を選択します。
SUSE では現在、`raw` および `qcow2` のイメージのみをサポート対象としています。また、後者は LVM ボリュームグループベースのプールの場合は選択できません。

次に［最大容量］を指定します。ここではディスクイメージが利用可能な最大のサイズを指定します。`qcow2` 形式を選択していない場合は、［割り当て］の容量も設定することができます。［割り当て］で指定した容量は初期の割り当てサイズとなり、［最大容量］と異なる値を設定すると、スパース形式のファイルが作成され、必要に応じてサイズが拡張されるようになります。`qcow2` 形式の場合は、［バッキングストア］(「バッキングファイル」と呼ぶ場合もあります) を使用することもできます。これはイメージのベースとなるファイルを指定するためのもので、新しく作成した `qcow2` イメージには、ベースからの変更点のみが記録されるようになります。

3. ボリュームの作成を開始するには［完了］を押します。

8.2.2.2.3 ストレージプールからのボリュームの削除

ボリュームの削除はストレージマネージャからのみ実施することができます。削除したいボリュームを選択して、［ボリュームの削除］を押します。確認メッセージが表示されたら［はい］を押してください。



警告: 使用中であってもボリュームは削除可能である件について

ボリュームは動作中の VM ゲスト から使用されている状況下でも削除することができます。削除したボリュームを復元する手段はありません。

対象のボリュームが VM ゲスト から使用されているかどうかを確認したい場合は、ストレージマネージャ内の［使用中］列を確認してください。

9 ゲストのインストール

改訂履歴

2025-06-12

VM ゲスト にはオペレーティングシステムやデータを含むイメージのほか、VM ゲスト の仮想的なハードウエアリソースを表す設定ファイルが存在しています。VM ゲスト は VM ホストサーバ 内で取り扱われ制御される仕組みであることから、本章では VM ゲスト をインストールする際の一般的な手順を説明しています。

仮想マシンには、インストールするオペレーティングシステムの要件が存在するほかには、特段の要件はほとんどありません。また、仮想マシンのホスト環境向けにオペレーティングシステムが最適化されていない場合、ハードウエア支援 による仮想化のみを完全仮想化モードで動作させる必要が生じるほか、特殊なデバイスドライバを読み込む必要も発生します。一方の VM ゲスト 側に提供されるハードウエアは、ホスト側の設定に従って決まります。

なお、複数の仮想マシンを作成してそれぞれでライセンス済みのオペレーティングシステムを動作させる場合は、そのライセンス構成にも注意してください。詳しくはオペレーティングシステムのライセンス同意書などの資料をお読みください。

9.1 GUI ベースのゲストインストール



ヒント: 新しい仮想マシンに対する既定のオプション設定の変更について

新しい仮想マシンを作成する際にはいくつかのオプションが自動的に設定されますが、この設定を変更することができます。たとえば新しい仮想マシンに対しては UEFI を使用するように設定したい場合は、仮想マシンマネージャ のメインメニューから［編集］›［設定］を選択して、［新しい仮想マシン］内で［UEFI］を選択します。



図 9.1: 新しい仮想マシンに対する既定のオプション設定

［新しい仮想マシン］ウイザードを利用することで、仮想マシンを作成し、オペレーティングシステムをインストールするまでに必要な作業を順に実施することができるようになっています。このウイザードを起動するには、仮想マシンマネージャ を起動したあと、［ファイル］›［新しい仮想マシン］を選択します。それ以外の方法としては、YaST を起動したあと［仮想化］›［Create Virtual Machines］を選択してもかまいません。

1. YaST もしくは 仮想マシンマネージャ で［新しい仮想マシン］ウイザードを起動します。

2. インストール元を選択します。ローカルに保存しておいたメディアか、ネットワーク上にあるインストール元を選択することができます。既存の VM ゲスト を取り込んで使用したい場合は、［既存のディスクイメージをインポート］を選択します。
   Xen ハイパーバイザが動作する VM ホストサーバ の場合、準仮想化 (paravirt) もしくは完全仮想化 (fullvirt) のいずれかを選択することができます。選択肢は［アーキテクチャオプション］内に表示されます。なお、選択の内容によっては、インストールオプションで選択できないものがあることもあります。

3. 直前の手順での選択内容に応じて、下記のデータを指定する必要があります:

   ［ローカルのインストールメディア (ISO イメージまたは CD-ROM ドライブ)］
   インストールデータを含む ISO イメージの VM ホストサーバ 内でのパスを指定します。libvirt のストレージプール内のボリュームとして利用できるように設定している場合は、［参照］を押して選択することもできます。詳しくは 第12章「高度なストレージ設定」をお読みください。
   上記以外にも、VM ホストサーバ 内の光学ドライブに CD-ROM や DVD のメディアが挿入されていれば、それを選択することもできます。

［ネットワークインストール (HTTP, HTTPS, or FTP)］

インストール元の［URL］を指定します。URL として指定可能なプロトコルには、`ftp://`，`http://`，`https://` などがあります。

［URL のオプション］内には、自動インストール用のファイル (AutoYaST や Kickstart など) を選択したり、カーネルのパラメータを指定したりすることができるオプションが用意されています。また、URL を指定した場合、通常はオペレーティングシステムを自動的に検出しますが、うまくいかない場合は手作業で指定することもできます。この場合は、［インストールメディアまたはソースから自動検出します］のチェックを外して、［OS タイプ］および［バージョン］に手動入力することもできます。

［既存のディスクイメージをインポート］

既存のイメージを利用して VM ゲスト の設定を行いたい場合は、まず VM ホストサーバ内でのイメージのパスを指定してください。なお、libvirt のストレージプール内のボリュームとして利用できるように設定している場合は、［参照］を押して選択することもできます。詳しくは 第12章「高度なストレージ設定」をお読みください。

［手動インストール］

このインストール方法は、仮想マシンのコンポーネントを手作業で設定し、後から OS をインストールしたい場合に適切です。仮想マシンを製品に合わせて調整したい場合は、`sles` 等のように OS の名称を入力したあと、表示された一覧の中からバージョンを選択してください。

4. 次に新しい仮想マシンに設定する、メモリサイズと CPU 数を指定します。

5. この手順は、［既存のディスクイメージをインポート］を選択した場合は省略されます。
VM ゲスト に対する仮想ハードディスクを設定します。新しいディスクイメージを作成するか、もしくはストレージプールから既存のものを選択 (詳しくは 第12章「高度なストレージ設定」をお読みください) して進めることができます。ディスクを作成するよう選択した場合は、`qcow2` 形式のイメージを作成します。また、既定では `/var/lib/libvirt/images` 内にイメージを配置します。
ディスクの設定は任意です。CD や DVD から直接実行することのできるライブシステムをお使いの場合は、［この仮想マシンにストレージデバイスを割り当てます］の選択を外して、ディスクの作成を行わないこともできます。

6. ウイザードの最後の画面では、仮想マシンに設定する名前を指定します。また、仮想マシンのハードウエア設定もカスタマイズすることができます。この場合は、［インストールの前に設定をカスタマイズする］を選択してください。また、［ネットワークの選択］では、ネットワークデバイスを選択することもできます。［Bridge device］を選択した場合は、ホスト側で設定されている最初のブリッジが自動的に選択されます。それ以外のブリッジを使用したい場合は、テキストボックス内にそのブリッジ名を入力してください。

［完了］を押します。

7. 直前の手順で既定値のまま進めた場合、ここでインストール処理が始まります。［インストールの前に設定をカスタマイズする］を選択した場合は、VM ゲスト の設定ダイアログが表示されます。VM ゲスト の設定に関する詳細は、第13章「仮想マシンマネージャ を利用した仮想マシンの設定」をお読みください。
設定が終わったら、［インストールの開始］を押してください。



ヒント: 仮想マシンに対する特殊キーの送信について

インストールが始まると、仮想マシンマネージャ のコンソールウインドウが表示されます。ただし、`Ctrl` – `Alt` – `F1` などの特殊なキー入力については VM ホストサーバ 側で解釈されてしまい、仮想マシンには送信されません。VM ホストサーバ ではなく仮想マシンにキー入力を送信したい場合は、「sticky key」と呼ばれる機能をお使いください。これは `Ctrl` , `Alt` , `Shift` を 3 回押下することで有効化されます。すると、直後のキー入力が、仮想マシンに送信されるようになります。

たとえば `Ctrl` – `Alt` – `F2` を Linux の仮想マシンに送信したい場合は、 `Ctrl` を 3 回押したあと、 `Alt` – `F2` を押してください。なお、 `Alt` を 3 回押したあとは、 `Ctrl` – `F2` を押します。

この sticky key 機能は、仮想マシンマネージャ で VM ゲスト をインストールしている際だけでなく、インストール後でも利用することができます。

9.1.1 PXE 起動向けの仮想マシンの設定

PXE 起動を使用することで、物理メディアやインストールディスクイメージを使用することなく、ネットワーク経由でインストールメディアにアクセスして起動することができるようになります。

PXE サーバから仮想マシンを起動するように設定するには、下記の手順を実施します:

1. 9.1項「GUI ベースのゲストインストール」に示されている手順でインストールウイザードを起動します。
2. ［手動インストール］を選択します。
3. ウイザードの最後では［インストールの前に設定をカスタマイズする］を選択しておきます。あとは［完了］を押します。
4. ［カスタマイズ］の画面が表示されたら、［ブートオプション］を選択します。
5. ［起動デバイスの順序］内にある［起動メニューを有効化する］を選択します。
   - ［VirtIO ディスク］が既定の起動オプションとして選択されていることを確認して［適用］を押します。

- 仮想マシンに対して PXE を既定値として設定したい場合は、下記を実施します:

  a. 起動デバイスの設定で NIC を選択してチェックを入れます。

  b. 右側のボタンで NIC を一番上に移動します。

  c. あとは［適用］を押すだけです。

6. ［インストールの開始］を押すと、インストールを始めることができます。画面が表示されたらすぐに `Esc` を押して、［1. iPXE］を選択してください。PXE サーバが正しく設定されていれば、PXE メニューが表示されるはずです。

9.2 `virt-install` によるコマンドラインからのインストール

`virt-install` は、`libvirt` ライブラリを利用した仮想マシンを作成することのできるコマンドラインツールです。グラフィカルなユーザインターフェイスを使用することのできない環境や、仮想マシンの作成を自動化したいような場合に有用です。

`virt-install` は多数のコマンドラインスイッチを含む、複雑なスクリプトです。下記に概要を示しますが、詳しい情報については `virt-install` (1) をお読みください。

一般的なオプション

- `--name ゲスト名`：新しく作成する仮想マシンの名前を指定します。名前は同じ接続のハイパーバイザ内で唯一のものでなければなりません。また、このゲスト名は設定ファイルの名前にもなるほか、後から `virsh` コマンドでこの名前を指定し、アクセスすることができます。英数字と `_-.:+` の文字を使用することができます。

- `--memory メモリサイズ`：仮想マシンに割り当てるメモリ量を、メガバイト単位で指定します。

- `--vcpus CPU_数`：仮想マシンに割り当てる CPU 数を指定します。性能を確保するため、仮想マシンのプロセッサ数の合計は、実際に搭載されているプロセッサ数と同じか、それより少なく設定しておくことをお勧めします。

仮想化の種類

- `--paravirt`：準仮想化のゲストを作成します。これは VM ホストサーバ が準仮想化と完全仮想化の両方に対応している場合の既定値となります。
- `--hvm`：完全仮想化のゲストを作成します。
- `--virt-type ハイパーバイザ名`：ハイパーバイザの種類を指定します。`kvm`，`xen` のいずれかを指定することができます。

ゲスト側のストレージ

`--disk`，`--filesystem`，`--nodisks` のいずれかのオプションを指定して、新しく作成する仮想マシンのストレージの種類を設定します。たとえば `--disk size=10` のように指定すると、ハイパーバイザの既定のイメージ配置先に、10 GB のディスクを作成し、VM ゲスト に割り当てます。`--filesystem VM_ホスト内でのパス` のように指定すると、指定した VM ホストサーバ内のディレクトリをゲストからアクセスできるようになります。`--nodisks` を指定すると、VM ゲスト にローカルストレージを割り当てない意味になります (ライブ CD などの用途に便利です) 。

インストール方法

`--location`，`--cdrom`，`--pxe`，`--import`，`--boot` のいずれかのオプションを指定して、インストール方法を指定します。

インストール環境へのアクセス

`--graphics 値` を指定することで、インストール環境へのアクセス方法を指定することができます。openSUSE Leap では、`vnc` もしくは `none` のいずれかの値をサポートしています。`virt-install` で VNC を指定した場合、`virt-viewer` を起動しようとします。このコマンドがインストールされていないか、実行することができない場合、VM ゲスト に対して手作業で接続を行ってください。`virt-install` でビューアを起動しないように明示的に指定したい場合は、`--noautoconsole` オプションを指定してください。VNC のセッションにアクセスするためのパスワードを指定したい場合は、`--graphics vnc,password=パスワード` のように指定してください。

`--graphics none` を指定した場合、VM ゲスト へのアクセスは、オペレーティングシステム側で提供されるサービス (例: SSH, VNC) を使用することになります。インストールシステムでこれらのサービスを有効化する方法について、詳しくはオペレーティングシステムのインストールマニュアルをお読みください。

カーネルと initrd ファイルの設定

ネットワークからのインストールなどでは、インストーラで使用するカーネルと initrd を直接指定することもできます。

起動時のパラメータを指定したい場合は、`--extra-args` オプションをお使いください。このパラメータでは、ネットワークの設定を行うこともできます。詳しくは https://ja.opensuse.org/SDB:Linuxrc をお読みください。

例 9.1: HTTP サーバからのカーネルと INITRD の読み込み

```
# virt-install --location \
"http://download.opensuse.org/pub/opensuse/distribution/leap/15.0/repo/oss" \
--extra-args="textmode=1" --name "Leap15" --memory 2048 --virt-type kvm \
--connect qemu:///system --disk size=10 --graphics vnc --network \
network=vnet_nated
```

コンソールの有効化

既定では、`virt-install` で新しく作成する仮想マシンに対して、コンソールは有効化されません。有効化したい場合は、`--extra-args="console=ttyS0 textmode=1"` のようにオプションを指定してください。たとえば下記のようになります:

```
> virt-install --virt-type kvm --name sles12 --memory 1024 \
 --disk /var/lib/libvirt/images/disk1.qcow2 --os-variant sles12
 --extra-args="console=ttyS0 textmode=1" --graphics none
```

インストールが完了したら、仮想マシン内の `/etc/default/grub` にある `GRUB_CMDLINE_LINUX_DEFAULT` の行に、`console=ttyS0` が設定されるようになります。

UEFI Secure Boot の使用



注記

SUSE では AMD64/Intel 64 の KVM ゲストに対してのみ UEFI Secure Boot のサポートを提供しています。Xen HVM ゲストでも UEFI ファームウエアに対応していますが、こちらは UEFI Secure Boot をサポートしていません。

規定では、`virt-install` を利用して新しい仮想マシンをインストールしようとすると、従来型の BIOS を利用するようになっています。UEFI を使用したい場合は、`--boot firmware=efi` を指定してください。この場合、UEFI Secure Boot に対応し、Microsoft 社の鍵を取り込み済みのファームウエアを選択します。Secure Boot を利用したくない場合は、`--boot firmware=efi,firmware.feature0.name=secure-boot,firmware.feature0.enabled=no` と指定してください。これにより、Secure Boot に対応しない UEFI ファームウエアを選択するようになります。
このほか、UEFI ファームウエアイメージを明示的に指定することもできます。仮想マシンで UEFI を使用する場合の高度な情報と設定例については、9.3.1項「高度な UEFI 設定」をお読みください。

例 9.2: `virt-install` コマンドラインの例

> 下記のコマンドライン例は、新しい SUSE Linux Enterprise 15 SP2 の仮想マシンを作成し、virtio で高速化したディスク環境と、ネットワークカードを接続する例です。ストレージとしては 10 GB の qcow2 形式のディスクイメージを作成し、インストール元のメディアはホスト側の CD-ROM ドライブを使用します。また、VNC のグラフィックも使用し、グラフィカルなフロントエンドを自動的に起動します。
>
> KVM
>
> ```
> > virt-install --connect qemu:///system --virt-type kvm \
> --name sle15sp2 --memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc
>  \
> --os-variant sle15sp2
> ```
>
> Xen
>
> ```
> > virt-install --connect xen:// --virt-type xen --hvm \
> --name sle15sp2 --memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc
>  \
> --os-variant sle15sp2
> ```

9.3 高度なゲストインストール手順

本章では、通常のインストール方法には含まれない機能、たとえば UEFI ファームウエアやメモリバルーンの手作業での設定、アドオン製品のインストールなどを使用する場合の手順について説明しています。

9.3.1 高度な UEFI 設定

仮想マシンで使用される UEFI ファームウエアは OVMF (Open Virtual Machine Firmware) が提供するものです。`qemu-ovmf-x86_64` パッケージには AMD64/Intel 64 の VM ゲストに対するファームウエアが含まれていますし、`qemu-uefi-aarch64` パッケージには AArch64 の VM ゲストに対するファームウエアが含まれています。どちらのパッケージにも複数のファームウエアが含まれていますが、それらはそれぞれ異なる機能が含まれています。このほかこれらのパッケージには、JSON ファームウエアディスクリプタファイルと呼ばれる、各ファームウエアが提供する機能を説明したファイルも含まれています。

`libvirt` では仮想マシンの UEFI ファームウエアの選択に際して、自動と手動の 2 種類の方式を提供しています。自動選択の場合、`libvirt` はユーザが指定したオプションセットに従ってファームウエアを選択します。明示的に何も機能を指定しない場合、`libvirt` は Secure Boot 対応で

Microsoft 社の鍵を取り込み済みのファームウエアを選択します。手動選択の場合、ファームウエアのフルパスを手作業で指定し、オプション機能については明示的に設定することになります。この場合、JSON ディスクリプタファイルを参照して、要件に合致するファームウエアを選択することができます。



ヒント

`/usr/share/qemu/firmware` ディレクトリには、`libvirt` が使用する全ての JSON ファイルが含まれています。このファイルには、各機能の情報を含むファームウエアの詳細が書かれています。

なお、`virt-install` を使用する場合、boot オプションに firmware=efi パラメータを指定することで、ファームウエアの自動選択機能が有効化されます。この場合、ファームウエアに求める／求めない機能を指定して自動選択を行います。たとえば下記の例では、UEFI Secure Boot が無効化されたファームウエアを自動選択するように指定しています。

```
> virt-install --connect qemu:///system --virt-type kvm \
--name sle15sp5 --memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc \
--boot firmware=efi,firmware.feature0.name=secure-boot,firmware.feature0.enabled=no \
--os-variant sle15sp5
```



注記

ファームウエアの自動選択によって VM ゲスト が使用するファームウエアが変わってしまわないようにするため、`libvirt` では VM ゲスト の XML 設定ファイル内に自動選択したファームウエアを記録するようにしています。これにより、ファームウエアの自動選択は 1 回だけ動作することになります。いったんファームウエアを自動選択したあとは、VM ゲスト の管理者が明示的に変更しない限り、手動でのファームウエア選択と同じように動作することになります。

また、手動でのファームウエア選択を行う場合は、loader と nvram のパラメータを使用します。loader は必須パラメータで nvram は任意パラメータとなります。nvram パラメータは、UEFI の変数ストアの保存先を指定します。たとえば下記の例では、Secure Boot が有効化されたファームウエアを手動で指定しています。

```
> virt-install --connect qemu:///system --virt-type kvm \
--name sle15sp5 --memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc \
--boot loader=/usr/share/qemu/ovmf-x86_64-smm-
code.bin,loader.readonly=yes,loader.type=pflash,loader.secure=yes,nvram.template=/usr/
share/qemu/ovmf-x86_64-smm-vars.bin \
--os-variant sle15sp5
```



注記

`libvirt` は UEFI ファームウエアの動作を変更することができません。たとえば UEFI Secure Boot が有効化されたファームウエアを使用している場合、loader.secure=no を指定しても、UEFI Secure Boot を無効化することはできません。また `libvirt` は、指定したファームウエアが指定した機能を提供するかどうかを確認します。たとえば loader.secure=no を指定して UEFI Secure Boot を無効化した状態で、UEFI Secure Boot の有効化されたファームウエアを指定すると、設定が拒否されます。

また、`qemu-ovmf-x86_64` パッケージには、複数の UEFI ファームウエアイメージが含まれています。たとえば下記のファイルは、いずれも SMM と UEFI Secure Boot が有効化されていますが、Microsoft, openSUSE, SUSE UEFI のそれぞれ異なる鍵が取り込まれているファームウエアです:

```
# rpm -ql qemu-ovmf-x86_64
[...]
/usr/share/qemu/ovmf-x86_64-smm-ms-code.bin
/usr/share/qemu/ovmf-x86_64-smm-ms-vars.bin
/usr/share/qemu/ovmf-x86_64-smm-opensuse-code.bin
/usr/share/qemu/ovmf-x86_64-smm-opensuse-vars.bin
/usr/share/qemu/ovmf-x86_64-smm-suse-code.bin
/usr/share/qemu/ovmf-x86_64-smm-suse-vars.bin
[...]
```

AArch64 アーキテクチャの場合、パッケージ名は `qemu-uefi-aarch32` になります:

```
# rpm -ql qemu-uefi-aarch32
[...]
/usr/share/qemu/aavmf-aarch32-code.bin
/usr/share/qemu/aavmf-aarch32-vars.bin
/usr/share/qemu/firmware
/usr/share/qemu/firmware/60-aavmf-aarch32.json
/usr/share/qemu/qemu-uefi-aarch32.bin
```

上記では `*-code.bin` ファイルが UEFI ファームウエアファイル、`*-vars.bin` ファイルがそれぞれ対応する変数ストアイメージになります。変数ストアイメージは仮想マシンごとの不揮発性ストアの雛型として使用されるものです。変数ストアイメージは仮想マシンの作成時に `/var/lib/libvirt/qemu/nvram/` 以下にコピーされ、その後はコピーしたファイルに対して書き込みを行います。`code` や `vars` を含まないファイルは単独 UEFI イメージと呼ばれますが、これらはいずれも仮想マシンのシャットダウンで UEFI 変数が消えてしまう構造のため、あまり使い道はありません。

また `*-ms*.bin` ファイルには、実在するハードウエアにも搭載されている UEFI CA 鍵が含まれています。そのため、これらは `libvirt` での既定値として指定されています。また `*-suse*.bin` には SUSE 社の鍵が取り込まれています。それ以外にも、事前には全く鍵を含まないファームウエアも存在しています。

OVMF に関する詳細は、http://www.linux-kvm.org/downloads/lersek/ovmf-whitepaper-c770f8c.txt (英語) をお読みください。

9.3.2 インストール時のアドオン製品の取り込み

openSUSE Leap 等のオペレーティングシステムでは、インストール時にアドオン製品の追加を行うことができます。SUSE Customer Center 経由でアドオン製品のインストールソースが提供されている場合は、VM ゲスト 側での設定は特に必要とはなりませんが、CD/DVD や ISO イメージとして提供されている場合、標準のインストールメディアとアドオン製品のイメージの両方を VM ゲスト に指定する必要があります。

GUI ベースのインストールを行っている場合は、ウイザードの最後の手順で［インストール前にオンラインリポジトリを追加する］を選択し、［ハードウェアを追加］›［ストレージ］を選択してアドオン製品の ISO イメージを追加してください。この場合、イメージのパスを指定し、［デバイスの種類］を［CD-ROM デバイス］に設定します。

コマンドライン経由でインストールを行っている場合は、`--cdrom` ではなく、`--disk` オプションで CD/DVD ドライブを設定する必要があります。最初に指定されたほうのデバイスが起動用に使用されます。たとえば下記のコマンドラインでは、SUSE Linux Enterprise Server 15 と SUSE Enterprise Storage 拡張を一括でインストールすることができます:

```
> virt-install \
 --name sles15+storage \
 --memory 2048 --disk size=10 \
 --disk /path/to/SLE-15-SP7-Full-ARCH-GM-media1.iso-x86_64-GM-DVD1.iso,device=cdrom \
 --disk /path/to/SUSE-Enterprise-Storage-VERSION-DVD-ARCH-Media1.iso,device=cdrom \
 --graphics vnc --os-variant sle15
```

10 基本的な VM ゲスト の管理

改訂履歴

2025-06-19

VM ゲスト の起動や停止など、ほとんどの管理作業はグラフィカルなアプリケーションである 仮想マシンマネージャ とコマンドラインである `virsh` の両方で実施することができます。ただし、VNC 経由でのグラフィカルコンソールへのアクセスは、グラフィカルユーザインターフェイスでのみアクセスすることができます。



注記: リモートの VM ホストサーバ 内にある VM ゲスト の管理について

VM ホストサーバ 内で動作させる場合、`libvirt` のツールである 仮想マシンマネージャ, `virsh` , `virt-viewer` の各ツールを利用することで、ホスト内で動作する VM ゲスト を管理することができます。ただし、リモートの VM ホストサーバ にある VM ゲスト を管理することもできます。この場合、`libvirt` にリモートから接続できるように設定する必要があります。詳しくは 第11章「接続と認可」をお読みください。

仮想マシンマネージャ でリモートのホストに接続するには、まずは 11.2.2項「仮想マシンマネージャ による接続の管理」での説明に従って、接続を設定する必要があります。また、`virsh` や `virt-viewer` でリモートのホストに接続する場合は、`-c` オプションで接続 URI を指定する必要があります (たとえば `virsh -c qemu+tls://saturn.example.com/system` や `virsh -c xen+ssh://` など) 。接続 URI の書式は、使用する接続の種類とハイパーバイザによって異なります。詳しくは 11.2項「VM ホストサーバ への接続」をお読みください。

なお、本章内でのコマンド例では、いずれも接続 URI を略しています。

10.1 VM ゲスト の一覧表示

VM ホストサーバ 内で `libvirt` を利用して管理されている全 VM ゲスト の一覧表示

10.1.1 仮想マシンマネージャ を利用した VM ゲスト の一覧表示

仮想マシンマネージャ のメインウインドウには、接続している VM ホストサーバ 内にある全ての VM ゲスト が一覧表示されています。それぞれの行にはマシンの名前のほか、状態 ([実行中] , [一時停止中] , [シャットオフ]) がアイコンとテキストで、そして CPU の使用率を表すバーも表示されています。

10.1.2 `virsh` による VM ゲスト の一覧表示

VM ゲスト の一覧を表示するには、`virsh list` コマンドを使用します:

実行中の全ゲストの表示

```
> virsh list
```

実行中／停止中両方の全ゲストの表示

```
> virsh list --all
```

詳細およびさらなるオプションについては、`virsh help list` または `man 1 virsh` で表示されるマニュアルページをお読みください。

10.2 コンソール経由での VM ゲスト へのアクセス

VM ゲスト は VNC 接続でアクセスすることができる (グラフィカルコンソールの場合) ほか、ゲスト側のオペレーティングシステムで対応している必要がありますが、シリアルコンソール経由でアクセスすることもできます。

10.2.1 グラフィカルコンソールの表示

VM ゲスト のグラフィカルコンソールを表示することで、物理マシンに VNC 接続しているのと同じように仮想マシンを扱うことができるようになります。また、接続先の VNC サーバで認証を求めるよう設定している場合、ユーザ名 (必要であれば) とパスワードの入力を求められます。

VNC コンソール内でマウスのボタンを押すと、マウスカーソルは「捕捉」状態になり、コンソールの外側には移動できなくなります。捕捉を解除するには、`Alt` – `Ctrl` を押してください。



ヒント: マウスカーソルの捕捉回避について

マウスカーソルがコンソール内に捕捉されず、VM ゲスト のウインドウの内側と外側を自由に行き来できるようにしたい場合は、VM ゲスト にタブレット入力デバイスを追加してください。詳しくは 13.5項「入力デバイス」をお読みください。

`Ctrl` – `Alt` – `Del` などの特殊なキー入力は、ホスト側で処理されてしまい、VM ゲスト には配送されません。これらの特殊なキー入力を VM ゲスト 側に受け渡すには、VNC ウインドウ内にある［キーの送信］メニューの中から、送信したいキーを選択してください。なお、［キーの送信］メニューは 仮想マ

シンマネージャと `virt-viewer` を利用している場合にのみ使用することができます。仮想マシンマネージャでは、ヒント: 仮想マシンに対する特殊キーの送信について で説明している手順で、「sticky key」を使用することもできます。



注記: 対応する VNC ビューアについて

仕様上は VNC に対応していれば、全てのビューアから VM ゲストに接続することができます。ただし、ゲストへのアクセスに際して SASL 認証を使用している場合や TLS/SSL 接続を設定しているような場合、利用できる VNC ビューアは限られてきます。`tightvnc` や `tigervnc` などの一般的な VNC ビューアには、SASL 認証の機能も、TLS/SSL 接続への対応も用意されていません。仮想マシンマネージャと `virt-viewer` の組み合わせ以外で唯一対応しているものは、Remmina (詳しくは『リファレンス』、第4章「VNC によるリモートグラフィカルセッション」、4.2項「Remmina: リモートデスクトップクライアント」をお読みください) のみとなります。

10.2.1.1 仮想マシンマネージャを利用したグラフィカルコンソールの表示

1. 仮想マシンマネージャ内にある VM ゲストの項目を選択して、マウスの右ボタンを押します。
2. ポップアップメニューから [開く] を選択します。

10.2.1.2 `virt-viewer` を利用したグラフィカルコンソールの表示

`virt-viewer` はシンプルな VNC ビューアで、VM ゲストのコンソール用にいくつかの機能が追加されています。たとえば「wait」モードを指定して起動すると、接続を行う前に VM ゲストが開始されるのを待つようになります。また、VM ゲストが再起動された場合、自動的に再接続する機能も用意されています。

`virt-viewer` では、VM ゲストの指定を名前のほか、ID や UUID で指定することができます。`virsh list --all` を実行すると、これらのデータを表示することができます。

実行中や一時停止中のゲストに接続する場合は、ID や UUID 、もしくは名前で選択することができます。シャットオフされている VM ゲストの場合は、ID が付与されていませんので、UUID や名前で選択してください。

ID `8` のゲストに接続:

```
> virt-viewer 8
```

`sles12` という名前の停止中のゲストに接続; ゲストの起動が行われるまで待機する設定:

```
> virt-viewer --wait sles12
```

`--wait` オプションを指定すると、VM ゲスト が実行中でない場合、接続が保留状態になります。ゲストが起動すると、ビューアが表示されるようになります。

詳しくは `virt-viewer --help` もしくは `man 1 virt-viewer` をお読みください。



注記: SSH 経由のリモート接続でのパスワード入力について

SSH 経由で接続しているホストに対して `virt-viewer` で接続を開く場合、SSH のパスワードは 2 回入力する必要があります。最初の 1 回は `libvirt` での認証に、もう 1 回は VNC サーバとの認証になります。なお、2 回目のパスワードは、virt-viewer の開始時にコマンドラインで指定する必要があります。

10.2.2 シリアルコンソールへの接続

仮想マシンのグラフィカルコンソールに接続するには、VM ゲスト にアクセスするクライアント側でも、グラフィカル環境を用意する必要があります。グラフィカルな環境を必要としない場合や、使用したくない場合は、`virsh` でシリアルコンソール経由を介してシェルにアクセスすることができます。VM ゲスト のシリアルコンソールにアクセスするには、下記のようなコマンドを入力して実行します:

```
> virsh console sles12
```

`virsh console` には 2 種類のオプションフラグが存在しています。`--safe` を指定すると、コンソールに対して排他アクセスを行おうとします。`--force` を指定すると、接続を行う前に既存の接続を全て切断します。いずれの機能とも、ゲスト側のオペレーティングシステムでの対応が必要となります。

VM ゲスト に対してシリアルコンソール経由でアクセスするには、ゲスト側のオペレーティングシステムがシリアルコンソールに対応し、適切に設定されている必要があります。詳しくはゲスト側のオペレーティングシステムのマニュアルをお読みください。



ヒント: SUSE Linux Enterprise や openSUSE のゲストに対するシリアルコンソールの有効化について

SUSE Linux Enterprise や openSUSE の場合、シリアルコンソールへのアクセスは既定で無効化されています。有効化したい場合は、下記のようにします:

SLES 12, 15 もしくは openSUSE

YaST ブートローダモジュールを起動して、［カーネルのパラメータ］タブに切り替えます。あとは［オプションのカーネルコマンドラインパラメータ］の欄に、`console=ttyS0` を追加します。

SLES 11

YaST ブートローダモジュールを起動して、シリアルコンソールを有効化したい起動項目を選択します。選択を行ったら［編集］を押し、［オプションのカーネルコマンドラインパラメータ］の欄に `console=ttyS0` を追加します。これに加えて、`/etc/inittab` ファイルを編集し、下記の行のコメント文字を外します:

```
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
```

10.3 VM ゲストの状態変更 (開始／停止／一時停止)

VM ゲストの開始や停止、一時停止は、仮想マシンマネージャもしくは `virsh` で行うことができます。また、VM ホストサーバの起動時に VM ゲストを自動的に開始するように設定することもできます。

なお、VM ゲストをシャットダウンする際は、正常な手順でシャットダウンを行うか、もしくは強制的にシャットダウンを行うかを選択することができます。強制的なシャットダウンはコンピュータの電源を抜く行為と同じであり、正常な手順でシャットダウンができない場合にのみ実施すべきものです。また、強制的なシャットダウンを行うと、VM ゲスト内のファイルシステムを破壊してしまうことがあるほか、データを失ってしまうこともあります。



ヒント: 正常なシャットダウン

正常にシャットダウンを行うには、VM ゲスト側が ACPI に対応するよう設定されていなければなりません。仮想マシンマネージャでゲストを作成している場合、VM ゲストの ACPI 対応は自動的に設定されます。

ゲスト側のオペレーティングシステムにも依存しますが、ACPI が利用できれば正常にシャットダウンできるというわけではありません。本番環境で使用する場合は、あらかじめ正常にシャットダウンできること、および再起動できることを確認してからご使用ください。なお、openSUSE や SUSE Linux Enterprise Desktop などでは、シャットダウンや再起動を行うのに Polkit による認可を必要とするように設定することもできます。このポリシーが全ての VM ゲスト で無効化されていることを確認してください。

Windows XP や Windows Server 2003 のゲストインストール時に ACPI を有効化した場合、VM ゲスト の設定のみを変更しただけでは設定不足となります。詳しくは下記をお読みください:

- https://support.microsoft.com/en-us/kb/314088
- https://support.microsoft.com/en-us/kb/309283

なお、VM ゲスト 側の設定にかかわらず、ゲスト側のオペレーティングシステム内からシャットダウンを行うことで、正常なシャットダウンを実現することができます。

10.3.1 仮想マシンマネージャ を利用した VM ゲスト の状態変更

VM ゲスト の状態変更は 仮想マシンマネージャ のメインウインドウのほか、VNC ウインドウからも行うことができます。

手順 10.1: 仮想マシンマネージャ ウインドウ内からの状態変更

1. VM ゲスト の項目を選択して、マウスの右ボタンを押します。
2. 表示されたポップアップメニューから、［実行］,［一時停止］のいずれか、もしくは［シャットダウン］内にある選択肢のいずれかを選択します。

手順 10.2: VNC ウインドウからの状態変更

1. 10.2.1.1項「仮想マシンマネージャ を利用したグラフィカルコンソールの表示」に示されている手順に従って、VNC ウインドウを表示します。
2. ツールバー内、もしくは［仮想マシン］メニュー内にある［実行］,［一時停止］のいずれか、もしくは［シャットダウン］内にある選択肢のいずれかを選択します。

10.3.1.1 VM ゲスト の自動起動

VM ホストサーバ の起動時にゲストを自動的に開始するように設定することができます。この機能は既定では有効化されておらず、それぞれの VM ゲスト に対して個別に設定する必要があります。全ての仮想マシンを一括で有効化することはできません。

1. 仮想マシンマネージャ で設定したい VM ゲスト の項目を選んでダブルクリックし、コンソールを表示します。
2. [表示] › [詳細] を選んで、VM ゲスト の設定ウインドウを表示します。
3. [ブートオプション] を選択し、表示された [ホスト起動時に仮想マシンを起動する] を選択します。
4. [適用] を押して新しい設定を保存します。

10.3.2 `virsh` を利用した VM ゲスト の状態変更

下記の例では、「sles12」という名前の VM ゲスト の状態を変更しています。

開始

```
> virsh start sles12
```

一時停止

```
> virsh suspend sles12
```

復元 (一時停止からの再開)

```
> virsh resume sles12
```

再起動

```
> virsh reboot sles12
```

正常なシャットダウン

```
> virsh shutdown sles12
```

強制シャットダウン

```
> virsh destroy sles12
```

自動開始の有効化

```
> virsh autostart sles12
```

自動開始の無効化

```
> virsh autostart --disable sles12
```

10.4 VM ゲスト の状態保存と状態復元

VM ゲスト の状態保存を行うことで、ゲストのメモリの内容を保存することができます。この操作はコンピュータの ハイバネーション (休止状態) に似た操作です。状態保存された VM ゲスト の復元を行うことで、その時点の状態に素早く戻すこともできます。

状態保存を行うと、VM ゲスト は一時停止状態になり、その時点でのメモリ内容をファイルに保存したあと、VM ゲスト が停止されます。ただし、この操作では VM ゲスト の仮想ディスクのコピーが作成されるわけではありません。また、仮想マシンの状態保存にかかる時間は、割り当てているメモリ量に依存して決まります。また状態保存を行うと、VM ゲスト に割り当てていたメモリが VM ホストサーバ 側に戻され、利用できるようになります。

状態復元の操作は、以前に状態保存しておいた VM ゲスト のメモリ状態を読み込んで、実行中の状態に戻す操作です。ここではゲストの起動処理は発生せず、状態を保存した時点の状態に戻します。この操作は、ハイバネーションからの復帰に似ています。

`libvirt` ではいくつかの状態保存形式に対応しています。既定では `raw` 形式が使われ、VM ゲスト のメモリページをそのまま連続したストリームに記録します。なお `raw` 形式は、複数の同時読み込み／同時書き込みには適切ではありません。

`raw` 状態保存形式に加え、`libvirt` では `zstd`，`lzop`，`gzip`，`bzip2`，`xz` の各形式に対応しています。これらは `raw` 形式と同様に連続したストリームに書き込みますが、それぞれの名前で示された圧縮アルゴリズムで圧縮処理が行われます。これによりファイルサイズが小さくなり容量の節約につながりますが、保存や復元にかかる時間は長くなり、ホストの CPU 資源もそれなりに使用してしまいます。

このほか `sparse` 状態保存形式では、事前に計算された固定のオフセット値を利用して VM ゲスト 内のメモリページの読み込み／書き込みを行います。保存されたファイルは VM ゲスト のメモリの論理サイズにおおよそ等しくなりますが、ディスク内で占有するサイズは VM ゲスト が実際に使用しているサイズになります。VM ゲスト のメモリページで固定のオフセット値を使用することで、`sparse` 形式は複数の同時読み込み／同時書き込みに対応できますので、特にメモリ割り当ての大きな VM ゲスト の保存や復元にかかる時間が削減できることになります。

既定の状態保存形式は、`/etc/libvirt/qemu.conf` 内の `save_image_format` で設定することができます。このほか、状態保存の実施時に `virsh` で指定することもできます。`virsh` による状態保存／状態復元については、10.4.2項「`virsh` による状態保存と状態復元」をお読みください。

なお、VM ゲスト の状態がファイルに保存されることから、処理を実施するにあたっては十分な容量を確保する必要があります。`sparse` 状態保存形式を使用した場合、保存ファイルの論理サイズはおおよそ VM ゲスト のメモリ割り当て量に等しくなります。ただし、ディスク内における実際のサイズは、VM ゲスト のメモリ使用状況に依存してそれより小さくなります。これは VM ゲスト 内で未使用のメモリをファイルに書き込まないためで、このような理由から `sparse` (疎 (まば) らな) という名前が付けられています。

`raw` 状態保存形式の場合は、論理サイズと実際のサイズは等しくなります。いずれも VM ゲスト のメモリ使用状況に依存して決まります。`raw` と `sparse` のどちらかを使用する場合、ファイルサイズは下記のコマンドで見積もることができます (メガバイト単位):

```
> free -mh | awk '/^Mem:/ {print $3}'
```

上記以外の圧縮形式の場合はこれより小さくなりますが、どれだけ小さくなるのかは圧縮アルゴリズムの効率次第です。



警告: 保存後の復元作業について

VM ゲスト の状態保存を行った後は、必ず状態復元を行うものとし、通常の起動や開始は行ってはなりません。通常の起動や開始を行ってしまうと、マシンの仮想ディスクが途中の状態のまま起動することになってしまうほか、状態保存のファイルとも整合性が取れなくなってしまうため、状態復元を行うとシステムが壊れてしまいます。

状態保存された VM ゲスト を正しく使用し続けるには、状態復元を忘れずに実施してください。また、`virsh` で VM ゲスト の状態保存を行った場合は、仮想マシンマネージャ を使用してはなりません。この場合は `virsh` で復元を行ってください。



重要: 復元後の VM ゲスト の時刻同期について

VM ゲスト の状態を保存してから長い時間 (2〜3時間以上) が経過した後に復元を行った場合、時刻同期サービスが正しく時刻を修正できなくなったりする場合があります。このような場合は、VM ゲスト の時刻を手作業で修正してください。たとえば KVM ホストであれば、QEMU のゲストエージェントを利用することで、`guest-set-time` コマンドでゲスト側の時刻を修正することができます。詳しくは 第21章「QEMU ゲストエージェント」をお読みください。

10.4.1 仮想マシンマネージャ を利用した状態保存と状態復元

手順 10.3: VM ゲスト の状態保存

1. VM ゲスト の VNC 接続ウインドウを表示します。また、ゲストが動作中であることを確認します。
2. [仮想マシン] › [シャットダウン] › [保存] を選択します。

手順 10.4: VM ゲスト の状態復元

1. VM ゲスト の VNC 接続ウインドウを表示します。また、ゲストが動作中でないことを確認します。
2. [仮想マシン] › [復元] を選択します。
   VM ゲスト を 仮想マシンマネージャ から状態保存していると、ゲストの [実行] オプションが表示されなくなります。ただし、警告: 保存後の復元作業について で説明しているとおり、`virsh` で状態保存を行っている場合に注意してください。

10.4.2 `virsh` による状態保存と状態復元

`libvirt` では、仮想マシンマネージャ よりもより細かく保存／復元処理を制御することができます。`virsh save` や `virsh restore` では様々な制御を行うことができますが、最も基本的な形式は VM ゲスト の名前や ID, UUID とファイル名を指定する形式となります。たとえば下記のようになります:

```
> virsh save openSUSE-Leap /virtual/saves/openSUSE-Leap.vmsav
```

VM ゲスト の復元の場合は、保存されたファイル名を指定するだけです。下記のようになります:

```
> virsh restore /virtual/saves/openSUSE-Leap.vmsav
```

VM ゲスト のメモリサイズが大きい場合で、特に高速なストレージシステムを使用しているような場合は、十分な転送速度を確保するために追加のオプション設定が必要となります。これは VM ホストサーバ のファイルシステムキャッシュが逆効果になってしまうためで、`bypass-cache` オプションを指定して無効化する必要があるためです。具体的には下記のようになります:

```
> virsh save --bypass-cache openSUSE-Leap /virtual/saves/openSUSE-Leap.vmsav
```

```
> virsh restore --bypass-cache /virtual/saves/openSUSE-Leap.vmsav
```

高速なストレージを搭載したシステムの場合、VM ゲスト のメモリページを同時に読み書きするように指定することで、保存や復元にかかる時間を改善することもできます。このような場合は、10.4項「VM ゲスト の状態保存と状態復元」にも示しているとおり、`sparse` 状態保存形式を指定して、複数の同

時読み込み／同時書き込みを行ってください。なお、同時数を指定するにあたっては、VM ホストサーバ 内で動作する他の処理に影響がないようにしてください。VM ホストサーバ のリソースが論理的に区切られているような場合であれば、一般的に VM ゲスト に割り当てられている物理 CPU 数と同じ値を指定してください。これは、保存処理の開始時に VM ゲスト の仮想 CPU が停止するため、それをそのまま保存処理に使用できることになるからです。

たとえば下記の例では、VM ゲスト の保存や復元を VM ホストサーバ のファイルシステムキャッシュを経由せず、同時 4 チャンネルで実施しています:

```
> virsh save --bypass-cache --image-format sparse --parallel-channels 4 openSUSE-Leap /
virtual/saves/openSUSE-Leap.vmsav
```

```
> virsh restore --bypass-cache --parallel-channels 4 /virtual/saves/openSUSE-Leap.vmsav
```

なお、状態保存時に使用したファイルの形式については、復元時に指定する必要はありません。

保存や復元の処理の詳細、および利用可能なオプションについての詳細は、`virsh help save`，`virsh help restore`，`man 1 virsh` をそれぞれお読みください。

10.5 スナップショットの作成と管理

VM ゲスト のスナップショット機能は、CPU やメモリ、デバイスの状態のほか、書き込み可能な全てのディスクの内容を含む、完全な仮想マシンのスナップショットです。仮想マシンのスナップショット機能を使用するには、接続されている全てのハードディスクが qcow2 形式である必要があるほか、少なくとも 1 つ以上のディスクが書き込み可能である必要があります。

スナップショットを使用することで、その時点でのマシンの状態を自由に復元できるようになります。これは特に、設定を誤ってしまった場合の巻き戻しや、多数のパッケージを誤ってインストールしてしまった場合の取り消しに便利な仕組みです。また、VM ゲスト がシャットオフ中に採取されたスナップショットを適用した場合は、適用後に起動を行う必要があります。また、スナップショットの適用を行うと、現時点での状態が破棄されます。



注記

スナップショット機能は KVM の VM ホストサーバ にのみ対応しています。

10.5.1 用語

スナップショットの種類を説明するにあたって、下記のいくつかの用語を使用しています:

内部スナップショット

スナップショットを元の VM ゲスト の qcow2 ファイル内に保存する方式です。このファイルには、スナップショットで保存された情報と、スナップショットを採取してからの変更点の両方が記録されます。内部スナップショットの利点としては、必要な全ての情報が 1 つのファイル内に保存されていて、マシン間で複製や移動を行う手間が省けるという点にあります。

外部スナップショット

外部スナップショットでは、元の qcow2 ファイルを保存して読み込み専用とし、それとは別に新しく qcow2 ファイルを作成してそこに変更点を記録します。元のファイルは バッキング もしくは ベース ファイルと呼ばれ、新しく作成したほうのファイルは オーバーレイ もしくは 派生 ファイルと呼ばれます。外部スナップショットは、VM ゲスト のバックアップを作成する際に便利です。ただし、外部スナップショットは 仮想マシンマネージャ ではサポートしておらず、`virsh` でも直接削除することができません。QEMU での外部スナップショットについて、詳しくは 35.2.4項「ディスクイメージの効率的な使用」をお読みください。

動作中スナップショット

VM ゲスト の動作中に採取するスナップショットを意味します。内部スナップショット形式での動作中スナップショットの場合、デバイスとメモリ、ディスクの各状態を保存することができます。`virsh` を利用した外部スナップショット形式での動作中スナップショットの場合は、メモリとディスクのうちいずれか、もしくはその両方の状態を保存することができます。

オフラインスナップショット

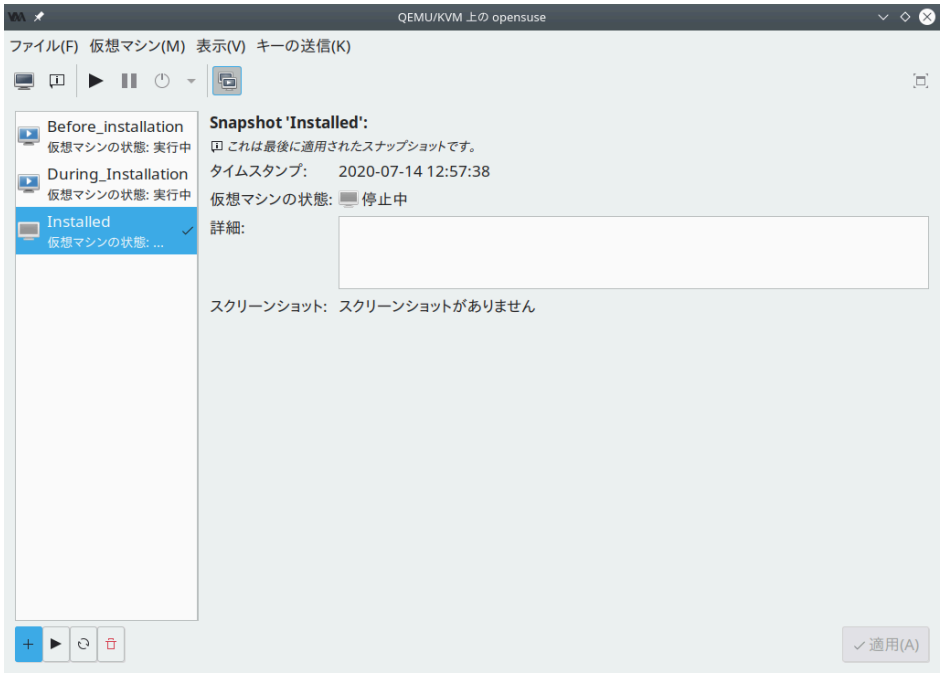
VM ゲスト がシャットオフ (シャットダウン) されている間に採取するスナップショットを意味します。ゲストが動作していない状態で採取することから、メモリも使用されていない状態になりますので、矛盾を一切発生させずに採取することができるようになります。

10.5.2 仮想マシンマネージャ を利用したスナップショットの作成と管理

重要: サポートは内部スナップショットのみである件について

仮想マシンマネージャ では動作中スナップショットであってもオフラインスナップショットであっても、内部スナップショットにしか対応していません。

仮想マシンマネージャ でスナップショットの管理ビューを表示するには、まず 10.2.1.1項「仮想マシンマネージャ を利用したグラフィカルコンソールの表示」の手順で VNC のウインドウを表示します。その後、［表示］›［スナップショット］を選択するか、ツールバーから［仮想マシンのスナップショットを管理］を選択します。



選択した VM ゲスト に対する既存のスナップショットの一覧が、ウインドウの左側に表示されます。現時点での最新のスナップショットには、緑色のチェックマークが付けられます。ウインドウの右側には、左側で選択しているスナップショットの詳細が表示されます。詳細にはスナップショットのタイトルとタイムスタンプのほか、採取時点での VM ゲスト の状態や説明などが表示されます。また、動作中スナップショットである場合には、その時点でのスクリーンショットも表示されます。なお、［詳細］の内容については、この表示から変更することができます。それ以外のスナップショットデータについては、変更できません。

10.5.2.1 スナップショットの作成

VM ゲスト に対して新しいスナップショットを採取するには、下記の手順を実施します:

1. オフラインスナップショットを採取する場合は、まず VM ゲスト をシャットダウンします。
2. VNC ウインドウ内の左下にある［新しいスナップショットを作成］ボタンを押します。［スナップショットを作成］ウインドウが表示されます。

3. ［名前］欄にスナップショットの名前を入力します。必要であれば、［詳細］欄に詳細を記述します。名前は作成後に変更することはできません。後からスナップショットの状況がわかるような名前を入力してください。
4. ［完了］を押すと採取が行われます。

10.5.2.2 スナップショットの削除

選択した VM ゲスト のスナップショットを削除するには、下記の手順を実施します:

1. VNC ウインドウ内の左下にある［選択したスナップショットを削除］ボタンを押します。
2. 削除の確認メッセージが表示されますので、［はい］を押します。

10.5.2.3 スナップショットの開始

選択したスナップショットで開始するには、下記の手順を実施します:

1. VNC ウインドウ内の左下にある［選択したスナップショットを実行］ボタンを押します。
2. 開始の確認メッセージが表示されますので、［はい］を押します。

10.5.3 `virsh` を利用したスナップショットの作成と管理

特定の VM ゲスト (下記では `admin_server`) に対するスナップショットの全一覧を表示するには、`snapshot-list` コマンドを使用します:

```
> virsh snapshot-list --domain admin_server
 名前                                作成時間                  状態
------------------------------------------------------------------------
 sleha_12_sp2_b2_two_node_cluster  2016-06-06 15:04:31 +0200 shutoff
 sleha_12_sp2_b3_two_node_cluster  2016-07-04 14:01:41 +0200 shutoff
 sleha_12_sp2_b4_two_node_cluster  2016-07-14 10:44:51 +0200 shutoff
 sleha_12_sp2_rc3_two_node_cluster 2016-10-10 09:40:12 +0200 shutoff
 sleha_12_sp2_gmc_two_node_cluster 2016-10-24 17:00:14 +0200 shutoff
 sleha_12_sp3_gm_two_node_cluster  2017-08-02 12:19:37 +0200 shutoff
 sleha_12_sp3_rc1_two_node_cluster 2017-06-13 13:34:19 +0200 shutoff
 sleha_12_sp3_rc2_two_node_cluster 2017-06-30 11:51:24 +0200 shutoff
 sleha_15_b6_two_node_cluster      2018-02-07 15:08:09 +0100 shutoff
 sleha_15_rc1_one-node             2018-03-09 16:32:38 +0100 shutoff
```

現時点での最新のスナップショットを表示するには、`snapshot-current` コマンドを使用します:

```
> virsh snapshot-current --domain admin_server
```

```
Basic installation incl. SMT for CLOUD4
```

特定のスナップショットに関する詳細を表示するには、`snapshot-info` コマンドを使用します:

```
> virsh snapshot-info --domain admin_server \
   -name  "Basic installation incl. SMT for CLOUD4"
名前:           Basic installation incl. SMT for CLOUD4
ドメイン:       admin_server
カレント:       はい (yes)
状態:           shutoff
場所:           内部
親:             Basic installation incl. SMT for CLOUD3-HA
子:             0
子孫:           0
メタデータ:     はい (yes)
```

10.5.3.1 内部スナップショットの作成

VM ゲスト に対して内部スナップショットを採取するには、それが動作中スナップショットであってもオフラインスナップショットであっても、`snapshot-create-as` コマンドを使用します:

```
> virsh snapshot-create-as --domain admin_server❶ --name "Snapshot 1"❷ \
--description "First snapshot"❸
```

❶ VM ゲスト の名前を指定します。必ず指定します。

❷ スナップショットの名前を指定します。後からスナップショットの状況がわかるような名前を入力してください。

❸ スナップショットの説明を指定します。指定しなくてもかまいません。

10.5.3.2 外部スナップショットの作成

`virsh` を使用することで、ゲストのメモリ状態またはディスク状態、もしくはその両方を外部スナップショットとして採取することができます。

ゲストのディスク状態の外部スナップショットを採取するには、それが動作中であってもオフラインであっても、`--disk-only` オプションを指定します:

```
> virsh snapshot-create-as --domain admin_server --name \
 "Offline external snapshot" --disk-only
```

`--diskspec` オプションを指定することで、外部スナップショットのファイル作成方法を制御することができます:

```
> virsh snapshot-create-as --domain admin_server --name \
 "Offline external snapshot" \
```

```
 --disk-only --diskspec vda,snapshot=external,file=/path/to/snapshot_file
```

ゲストのメモリ状態の外部スナップショットを採取するには、`--live` および `--memspec` オプションを指定します:

```
> virsh snapshot-create-as --domain admin_server --name \
 "Offline external snapshot" --live \
 --memspec snapshot=external,file=/path/to/snapshot_file
```

ゲストのディスクとメモリ状態の両方を外部スナップショットとして採取するには、`--live`，`--diskspec`，`--memspec` の各オプションを組み合わせて指定します:

```
> virsh snapshot-create-as --domain admin_server --name \
 "Offline external snapshot" --live \
 --memspec snapshot=external,file=/path/to/snapshot_file
 --diskspec vda,snapshot=external,file=/path/to/snapshot_file
```

詳しくは `man 1 virsh` 内の SNAPSHOT COMMANDS セクション (英語) をお読みください。

10.5.3.3 スナップショットの削除

外部スナップショットは `virsh` では削除できません。VM ゲスト の内部スナップショットを削除し、占有していたディスク領域を解放するには、`snapshot-delete` コマンドを使用します:

```
> virsh snapshot-delete --domain admin_server --snapshotname "Snapshot 2"
```

10.5.3.4 スナップショットの開始

指定したスナップショットで開始するには、`snapshot-revert` コマンドを使用します:

```
> virsh snapshot-revert --domain admin_server --snapshotname "Snapshot 1"
```

現在のスナップショット (VM ゲスト を最後にシャットダウンした状態) を開始する場合は、スナップショットの名前ではなく、`--current` オプションを指定すれば十分です:

```
> virsh snapshot-revert --domain admin_server --current
```

10.6 VM ゲスト の削除

既定では、`virsh` による VM ゲスト の削除では、XML 形式の設定ファイルのみを削除します。接続されているストレージは既定では削除されませんので、他の VM ゲスト に接続しなおして使用することもできます。仮想マシンマネージャ を使用する場合は、同時にストレージファイルも削除することができます。

10.6.1 仮想マシンマネージャ を利用した VM ゲスト の削除

1. 仮想マシンマネージャ 内にある VM ゲスト の項目を選択して、マウスの右ボタンを押します。
2. 表示されたコンテキストメニューから、［削除］を選択します。
3. 確認メッセージが表示されますので、再度［削除］ボタンを押します。これにより、VM ゲスト を恒久的に削除することができます。また、削除は取り消すことができません。
   仮想ディスクについても同時に削除したい場合は、［関連するストレージファイルを削除する］を選択します。こちらについても削除を取り消すことはできません。

10.6.2 `virsh` を利用した VM ゲスト の削除

VM ゲスト を削除するには、まずシャットダウンしておく必要があります。動作中の場合、ゲストを削除することができません。シャットダウンに関する情報については、10.3項「VM ゲスト の状態変更 (開始／停止／一時停止)」をお読みください。

`virsh` を使用して VM ゲスト を削除するには、`virsh undefine VM_名` のように入力して実行します。

```
> virsh undefine sles12
```

接続されているストレージファイルを自動的に削除するオプションは用意されていません。libvirt でストレージファイルを管理している場合は、8.2.1.4項「ストレージプールからのボリュームの削除」の手順に従って削除を行ってください。

10.7 監視

10.7.1 仮想マシンマネージャ を利用した監視

仮想マシンマネージャ を起動して VM ホストサーバ に接続すると、動作中のゲストの CPU 使用率が表示されます。

このツールでは、ディスクやネットワークの使用率に関する情報も表示することができます。ただし、あらかじめ［設定］で機能を有効化しておかなければなりません:

1. `virt-manager` を起動します。
2. ［編集］› ［設定］を選択します。

3. ［全般］から［ポーリング］のタブに切り替えます。
4. 悲痛に応じて、監視したい項目を選択します。それぞれ［ディスク I/O の取得］,［ネットワーク I/O の取得］,［メモリーの統計を取得する］です。
5. また、必要であれば［状態の更新間隔］の値も調整します。
6. ［設定］ダイアログを閉じます。
7. あとは［表示］›［グラフ］以下にある項目を選択して、グラフを表示させてください。

すると、ディスクやネットワークの統計情報が、仮想マシンマネージャ のメインウインドウ内に表示されるようになります。

より詳しいデータを閲覧したい場合は、VNC ウインドウからご確認ください。VNC ウインドウの表示方法は 10.2.1項「グラフィカルコンソールの表示」で説明しています。ツールバーから［仮想マシンの情報を表示］を押すか、もしくは［表示］›［詳細］を選択してください。統計情報は左側のメニュー内にある［性能］を選択すると、表示されるようになります。

10.7.2 `virt-top` を利用した監視

`virt-top` は、よく知られたプロセス監視ツールである `top` に似たコマンドラインツールです。`virt-top` は libvirt を利用して、さまざまなハイパーバイザで動作する VM ゲスト の統計情報を表示することができます。`xentop` のようなハイパーバイザ固有のツールではなく、`virt-top` のような汎用ツールの使用をお勧めします。

既定では、`virt-top` は実行中の全ての VM ゲスト に対する統計情報を表示します。ここにはメモリの使用率（`%MEM`）のほか、CPU の使用率（`%CPU`）とゲストの動作時間（`TIME`）が表示されます。データは定期的に自動更新されます (既定では 3 秒間隔で更新されます) 。下記の例では、VM ホストサーバ 内に合計 7 つの VM ゲスト が存在し、それらのうちの 4 つが停止されている状況を示しています:

```
virt-top 13:40:19 - x86_64 8/8CPU 1283MHz 16067MB 7.6% 0.5%
7 domains, 3 active, 3 running, 0 sleeping, 0 paused, 4 inactive D:0 O:0 X:0
CPU: 6.1%  Mem: 3072 MB (3072 MB by guests)

   ID S RDRQ WRRQ RXBY TXBY %CPU %MEM    TIME   NAME
    7 R  123    1  18K  196  5.8  6.0   0:24.35 sled12_sp1
    6 R    1    0  18K    0  0.2  6.0   0:42.51 sles12_sp1
    5 R    0    0  18K    0  0.1  6.0  85:45.67 opensuse_leap
    -                                           (Ubuntu_1410)
    -                                           (debian_780)
    -                                           (fedora_21)
    -                                           (sles11sp3)
```

既定での並び順は ID 順です。下記のキー入力を行うことで、並び順を変更することができます:

`Shift` – `P` ：CPU の使用率
`Shift` – `M` ：ゲストに割り当てているメモリ量
`Shift` – `T` ：時間
`Shift` – `I` ：ID

その他の情報をもとに並べ替えを行いたい場合は、 `Shift` – `F` を押して、表示される一覧から項目を選択してください。並び順を逆にしたい場合は、 `Shift` – `R` を押します。

`virt-top` では、VM ゲスト のデータをさまざまな形で表示することができます。これらは下記のキー入力を行うことで、即時に変更することができます:

`0` ：既定の表示
`1` ：物理 CPU の表示
`2` ：ネットワークインターフェイスの表示
`3` ：仮想ディスクの表示

`virt-top` には表示を変更するためのさまざまなキー入力や、プログラムの動作を変更するためのさまざまなコマンドラインが用意されています。詳しくは `man 1 virt-top` をお読みください。

10.7.3 `kvm_stat` を利用した監視

`kvm_stat` は、KVM の性能イベントを追跡する際に使用するツールです。`/sys/kernel/debug/kvm` を監視する仕組みであるため、まずは debugfs をマウントする必要があります。openSUSE Leap では既定でマウントされるように設定されていますが、何らかの理由でマウントされていない場合は、下記のように実行してマウントしてください:

```
> sudo mount -t debugfs none /sys/kernel/debug
```

`kvm_stat` は、下記に示す 3 種類の動作モードが用意されています:

```
kvm_stat                    # 1 秒間隔で更新
kvm_stat -1                 # 1 秒間情報採取して出力
kvm_stat -l > kvmstats.log  # ログ形式で 1 秒間隔で更新
                            # (表計算プログラムなどに取り込むことができます)
```

例 10.1: `kvm_stat` の出力例

```
kvm statistics

 efer_reload                   0       0
 exits                  11378946  218130
```

```
fpu_reload                62144     152
halt_exits               414866     100
halt_wakeup              260358      50
host_state_reload        539650     249
hypercalls                    0       0
insn_emulation          6227331  173067
insn_emulation_fail           0       0
invlpg                   227281      47
io_exits                 113148      18
irq_exits                168474     127
irq_injections           482804     123
irq_window                51270      18
largepages                    0       0
mmio_exits                 6925       0
mmu_cache_miss            71820      19
mmu_flooded               35420       9
mmu_pde_zapped            64763      20
mmu_pte_updated               0       0
mmu_pte_write            213782      29
mmu_recycled                  0       0
mmu_shadow_zapped        128690      17
mmu_unsync                   46      -1
nmi_injections                0       0
nmi_window                    0       0
pf_fixed                1553821     857
pf_guest                1018832     562
remote_tlb_flush         174007      37
request_irq                   0       0
signal_exits                  0       0
tlb_flush                394182     148
```

これらの値の解釈方法について、詳しくは https://clalance.blogspot.com/2009/01/kvm-performance-tools.html (英語) をお読みください。

11 接続と認可

改訂履歴

2024-06-27

多くの VM ゲスト が存在する複数の VM ホストサーバ を扱うようになると、管理に手間がかかるようになってしまいます。`libvirt` では、複数の VM ホストサーバ に対して一括で接続し、単一のインターフェイスから全ての VM ゲスト を管理することができるほか、それらのグラフィカルインターフェイスにも簡単に接続できるようになっています。

不正な接続をされないようにする目的で、`libvirt` ではさまざまな種類の接続 (TLS, SSH, Unix ソケット, TCP) に対応しています。これらでは、さまざまな認可メカニズム (ソケット, Polkit, SASL, Kerberos) を組み合わせて使用することができます。

11.1 認証

VM ゲスト の管理やそれらのグラフィカルコンソールへのアクセスは、正当なユーザからのアクセスのみを受け入れるように設定する必要があります。このような目的を達成するため、VM ホストサーバ 側では下記のような認証技術を使用することができます:

- パーミッションやグループの所有権を利用した、Unix ソケット向けのアクセス制御。この方式は、`libvirtd` の接続にのみ適用することができます。
- Polkit を利用した Unix ソケット向けのアクセス制御。この方式は、ローカルの `libvirtd` 接続にのみ適用することができます。
- SASL (Simple Authentication and Security Layer) を利用したユーザ名とパスワードによる認証。この方式は `libvirtd` と VNC のどちらの接続でも利用することができます。SASL での認証は、システムとは異なるユーザデータベースを利用して管理を行うことから、サーバ内にユーザを作成する必要はありません。また、SASL への認証接続は暗号化されます。
- Kerberos 認証。この方式は、`libvirtd` の接続にのみ適用することができます。また、本マニュアルでは説明していません。詳しくは https://libvirt.org/auth.html#ACL_server_kerberos をお読みください。
- 単独パスワード認証。この方式は、VNC 接続にのみ適用することができます。

重要: libvirtd への認証と VNC への認証は個別に設定する必要がある件について

VM ゲスト の管理機能 (`libvirtd` 経由) へのアクセスとグラフィカルコンソールへのアクセスは、それぞれ別々に設定する必要があります。管理ツールへのアクセスを制限しても、その設定は VNC 接続への設定に自動的に適用されることは ありません 。

TLS/SSL 接続を介してリモートから VM ゲスト にアクセスする場合、クライアント側で使用する証明書の鍵ファイルの読み込み権限を特定のグループに限定することで、間接的にアクセスを制御することができます。詳しくは 11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」をお読みください。

11.1.1 libvirtd の認証

`libvirtd` の認証は `/etc/libvirt/libvirtd.conf` で設定します。ここでの設定は、仮想マシンマネージャ や `virsh` など、全ての `libvirt` ツールに適用されます。

`libvirt` では 2 種類のソケットを提供しています。1 つは監視用の読み込み専用ソケット、もう 1 つは管理操作に使用するための読み書き可能なソケットです。それぞれのソケットに対する設定は、個別に行うことができます。既定のアクセス許可設定では、読み書き可能なソケットが `root` にのみ許可されるように制限 (`0700`) され、読み込み専用のソケットは誰にでもアクセスできるように公開 (`0777`) されています。

下記の手順では、読み書き可能なソケットに対するアクセス許可設定方法を示しています。読み込み専用ソケットに対しても同じような手順で実施することができます。また、設定作業は VM ホストサーバ内で行います。

注記: openSUSE Leap における既定の認証設定について

openSUSE Leap での既定の認証方式は、Unix ソケットに対するアクセス制御です。`root` のみに対してアクセスを許可しています。VM ホストサーバ で root 以外のユーザが `libvirt` ツールを使用しようとすると、Polkit を介して `root` のパスワード入力を求められます。パスワードを正しく入力することで、現在および将来使用できるようにアクセスが許可されます。

それ以外の方法としては、非特権ユーザに対して `libvirt` の「システム」アクセスを許可する方法があります。詳しくは 11.2.1項「非特権ユーザに対する「システム」アクセス」をお読みください。

推奨される認可方式

ローカル接続

- 11.1.1.2項「Polkit を利用した Unix ソケット向けのローカルアクセス制御」
- 11.1.1.1項「パーミッションとグループの所有権を利用した Unix ソケット向けのアクセス制御」

SSH 経由でのリモートトンネル

11.1.1.1項「パーミッションとグループの所有権を利用した Unix ソケット向けのアクセス制御」

リモート TLS/SSL 接続

- 11.1.1.3項「SASL を利用したユーザ名とパスワードによる認証」
- 無し (証明書へのアクセスを制限することで、クライアント側でアクセス制御)

11.1.1.1 パーミッションとグループの所有権を利用した Unix ソケット向けのアクセス制御

`root` 以外のユーザに対してアクセスを許可するには、特定のグループがソケットを所有し、アクセスできるように設定する必要があります (下記の例では、`libvirt` グループに対して許可する例を示しています) 。この認証方式は、ローカルとリモートの SSH 接続で使用することができます。

1. グループが存在していない場合は、下記を実行してソケットを所有すべきグループを作成します:

```
> sudo groupadd libvirt
```

> **重要: グループの存在について**
>
> このグループは、`libvirtd` の再起動を行うまでに存在していなければなりません。存在していない場合、再起動が失敗します。

2. あとはグループに対してユーザを追加していきます:

```
> sudo usermod --append --groups libvirt tux
```

3. `/etc/libvirt/libvirtd.conf` ファイル内の設定を下記のように変更します:

```
unix_sock_group = "libvirt"❶
unix_sock_rw_perms = "0770"❷
auth_unix_rw = "none"❸
```

❶ 所有グループを指定しています `libvirt` 。

❷ ソケットに対するアクセス許可を指定しています (`srwxrwx---`) 。

❸ その他の認証方式 (Polkit や SASL) を無効化しています。アクセスはソケットのパーミッションのみで処理されることになります。

4. `libvirtd` を再起動します:

```
> sudo systemctl start libvirtd
```

11.1.1.2 Polkit を利用した Unix ソケット向けのローカルアクセス制御

Polkit を利用した Unix ソケット向けのアクセス制御は、ローカルからの接続における openSUSE Leap の既定の認証方式です。そのため、`libvirt` 側の設定は不要です。Polkit の認証方式を有効化していれば、両方のソケットに対するパーミッションは `0777` に設定され、ソケットにアクセスしようとするそれぞれのアプリケーション側で、Polkit による認証が必要になります。

重要: Polkit の認証はローカル接続にのみ対応する件について

Polkit はリモートに対する認証には対応していませんので、VM ホストサーバ 自身からのローカル接続にのみ使用することができます 。

`libvirt` のソケットへのアクセスに対しては、2 種類のポリシーが存在しています:

- org.libvirt.unix.monitor : 読み込み専用ソケットに対するアクセス制御
- org.libvirt.unix.manage : 読み書き可能なソケットに対するアクセス制御

既定では、読み書き可能なソケットに対するアクセスのポリシーは、`root` のパスワード入力を一度だけ求め、その時点およびその後のセッションでは、そのまま権限を許可する設定になっています。

`root` のパスワードを入力させることなく、一般ユーザからソケットにアクセスできるようにするには、`/etc/polkit-1/rules.d` 内にルールを作成する必要があります。たとえば `/etc/polkit-1/rules.d/10-grant-libvirt` ファイルを下記の内容で作成すると、`libvirt` グループに所属する全てのメンバーに対して、ソケットへのアクセスを許可するようになります:

```
polkit.addRule(function(action, subject) {
  if (action.id == "org.libvirt.unix.manage" && subject.isInGroup("libvirt")) {
    return polkit.Result.YES;
  }
});
```

11.1.1.3 SASL を利用したユーザ名とパスワードによる認証

SASL はユーザ名とパスワードによる認証を提供する仕組みで、データの暗号化 (既定では digest-md5) にも対応しています。SASL は独自のユーザデータベースを利用して認証を行う仕組みであるため、VM ホストサーバ 内にユーザを作成する必要はありません。また、SASL は TCP 接続を利用し、TLS/SSL にも対応しています。



重要: digest-md5 暗号化による純粋な TCP と SASL による認証について

他の方法で TCP の接続が暗号化されていない限り、digest-md5 暗号化による認証を行っても、本番環境では十分なセキュリティであるとは言えません。このような構成は、テスト環境でのみ使用することをお勧めします。



ヒント: TLS/SSL を利用した SASL 認証について

TLS/SSL によるリモートからのアクセスを許可する場合、クライアント側で 証明書の鍵ファイルのアクセスを制限することで、間接的にアクセスを制御することができます。ただし、多くのクライアントを設定する場合、設定ミスを引き起こしやすい構成でもあります。TLS による認証を行う際は、SASL による認証も同時に行うものとして、サーバ側でも追加の制御を行うようにしてください。

SASL 認証を設定するには、下記の手順を実施します:

1. `/etc/libvirt/libvirtd.conf` ファイル内の設定を下記のように変更します:

   a. TCP 接続による SASL を使用したい場合:

   ```
   auth_tcp = "sasl"
   ```

   b. TLS/SSL 接続による SASL を使用したい場合:

   ```
   auth_tls = "sasl"
   ```

2. `libvirtd` を再起動します:

   ```
   > sudo systemctl restart libvirtd
   ```

3. libvirt の SASL 設定は `/etc/sasl2/libvirtd.conf` ファイル内に書かれています。通常は既定で設定されている内容を変更する必要はありませんが、TLS と SASL を併用する場合、不要なオーバーヘッドを削減するため、セッション暗号化を無効化 (TLS の時点で既に暗号化され

ているため) することができます。これは、`mech_list` の行をコメントアウトすることで実現することができます。ただし、下記の設定は TLS/SASL を併用する場合にのみ設定するものし、通常の TCP 接続では digest-md5 のままにしなければなりません。

```
#mech_list: digest-md5
```

4. 既定では SASL ユーザは何も設定されていませんので、誰もログインできない状態になります。下記のコマンドを利用して、ユーザを管理してください:

   `tux` というユーザを追加:

   ```
   saslpasswd2 -a libvirt tux
   ```

   `tux` というユーザを削除:

   ```
   saslpasswd2 -a libvirt -d tux
   ```

   ユーザー覧の表示:

   ```
   sasldblistusers2 -f /etc/libvirt/passwd.db
   ```



ヒント: `virsh` と SASL の認証の関係について

SASL 認証を使用している場合、`virsh` コマンドで接続を行うと、毎回ユーザ名とパスワードを尋ねられるようになります。`virsh` をシェルモードで動作させると、この手間を省くことができます。

11.1.2 VNC 認証

VM ゲスト のグラフィカルコンソールへのアクセスは `libvirt` の管理範囲外であり、ハイパーバイザ側で管理されているものであることから、VNC 認証については必ず設定しておく必要があります。メインとなる設定ファイルは `/etc/libvirt/<ハイパーバイザ名>.conf` です。本章では QEMU/KVM ハイパーバイザを説明していますので、設定ファイルは `/etc/libvirt/qemu.conf` になります。



注記: Xen 向けの VNC 認証について

KVM とは異なり、Xen の VM にはパスワードによる認証しか設定できません。詳しくは下記にある `libvirt` の設定オプションのうち、`<graphics type='vnc'...` の箇所をご覧ください。

VNC の認証は 2 種類の認証方式 (SASL, 単独パスワード認証) に対応しています。`libvirt` 側で SASL 認証を使用している場合、VNC 認証でも同じものを使用することをお勧めします。もちろん同じデータベースを使用するように設定することができます。

3 種類目のアクセス制限方式として、VNC サーバでの TLS 暗号化の使用という方式もあります。この場合、VNC クライアント側に x509 クライアント証明書を配置することになります。これらの証明書へのアクセスを制限することで、クライアント側で間接的にアクセスを制限することになります。詳しくは 11.3.2.4.2項「VNC over TLS/SSL: クライアント設定」をお読みください。

11.1.2.1 SASL を利用したユーザ名とパスワードによる認証

SASL はユーザ名とパスワードによる認証を提供するだけでなく、データの暗号化の機能も提供します。また、SASL は独自のデータベースを使用する方式であるため、VM ホストサーバ 内にユーザを作成しておく必要もありません。`libvirt` に対して SASL の認証を設定する場合、TLS/SSL 接続を利用することもできます。これらの接続の設定方法について、詳しくは 11.3.2.4.2項「VNC over TLS/SSL: クライアント設定」をお読みください。

VNC 向けに SASL 認証を設定するには、下記の手順を実施します:

1. まずは SASL の設定ファイルを作成します。作成にあたっては、既存の `libvirt` 向けファイルを使用することをお勧めします。既に `libvirt` 向けに SASL を設定してある環境で、ユーザ名とパスワードを含む全ての設定を共通で使用したい場合は、下記のようにしてシンボリックリンクを作成するだけでかまいません:

   ```
   > sudo ln -s /etc/sasl2/libvirt.conf /etc/sasl2/qemu.conf
   ```

   VNC にのみ SASL を設定する場合や、VNC 専用に個別の設定を作成したい場合は、既存のファイルをコピーして雛形として使用することをお勧めします:

   ```
   > sudo cp /etc/sasl2/libvirt.conf /etc/sasl2/qemu.conf
   ```

   あとは必要に応じて内容を修正してください。

2. `/etc/libvirt/qemu.conf` にある設定ファイルを、下記のように修正します:

   ```
   vnc_listen = "0.0.0.0"
   vnc_sasl = 1
   sasldb_path: /etc/libvirt/qemu_passwd.db
   ```

   最初のパラメータは、VNC サービスが待ち受けるべきアドレスを指定するものです。この場合、ローカルホストだけでなく、全てのアドレスから接続できる設定になります。2 つ目のパラメータは、SASL 認証を有効化するための設定です。

3. 既定では SASL ユーザは何も設定されていませんので、誰もログインできない状態になります。下記のコマンドを利用して、ユーザを管理してください:

   `tux` というユーザを追加:

   ```
   > saslpasswd2 -f /etc/libvirt/qemu_passwd.db -a qemu tux
   ```

   `tux` というユーザを削除:

   ```
   > saslpasswd2 -f /etc/libvirt/qemu_passwd.db -a qemu -d tux
   ```

   ユーザー覧の表示:

   ```
   > sasldblistusers2 -f /etc/libvirt/qemu_passwd.db
   ```

4. `libvirtd` を再起動します:

   ```
   > sudo systemctl restart libvirtd
   ```

5. 設定を変更する前から VM ゲスト を動作させていた場合は、それら全てを再起動します。再起動を行わない場合、VNC では SASL を使用することができません。



注記: 対応する VNC ビューアについて

現時点では、SASL 認証は 仮想マシンマネージャ と `virt-viewer` でのみ対応しています。これらのビューアは、TLS/SSL 接続にも対応しています。

11.1.2.2 パスワード単独認証

VNC サーバへのアクセスは、パスワード単独での認証で制限することもできます。この場合、全ての VM ゲスト に対してグローバルパスワードを設定するか、それぞれのゲストに対して個別にパスワードを設定することができます。なお、後者は VM ゲスト の設定ファイル内に記述します。



注記: グローバルパスワードの設定について

パスワード単独で認証を設定する場合、それぞれの VM ゲスト に対してパスワードを設定している場合でも、グローバルパスワードを設定しておくことをお勧めします。これにより、マシンごとのパスワードを設定し忘れた場合でも、代替となるパスワードが設定されることになりますので、最低限の保護は実現することができます。なお、グローバルパスワードは、マシンに対してそれ以外のパスワードが設定されていない場合にのみ適用されます。

手順 11.1: グローバル VNC パスワードの設定

1. `/etc/libvirt/qemu.conf` にある設定ファイルを、下記のように修正します:

```
vnc_listen = "0.0.0.0"
vnc_password = "パスワード"
```

   最初のパラメータは、VNC サービスが待ち受けるべきアドレスを指定するものです。この場合、ローカルホストだけでなく、全てのアドレスから接続できる設定になります。2 つ目のパラメータはパスワードです。パスワードは最大で 8 文字まで設定することができます。

2. `libvirtd` を再起動します:

```
> sudo systemctl restart libvirtd
```

3. 設定を変更する前から VM ゲスト を動作させていた場合は、それら全てを再起動します。再起動を行わない場合、VNC ではパスワード認証を使用することができません。

手順 11.2: VM ゲスト 固有の VNC パスワードの設定

1. `/etc/libvirt/qemu.conf` にある設定ファイルを下記のように修正し、ローカルホストだけでなく、全てのアドレスから接続できるようにします:

```
vnc_listen = "0.0.0.0"
```

2. 次にエディタを利用して、VM ゲスト の XML 設定ファイルを開きます。下記の `VM_名` には、実際の VM ゲスト の名前を入れてください。また、起動されるエディタは `$EDITOR` で指定されたエディタになります。何も値を設定していない場合、`vi` エディタを使用します。

```
> virsh edit VM_名
```

3. `type='vnc'` という属性が書かれた `<graphics>` タグを探します。たとえば下記のように書かれているはずです:

```
<graphics type='vnc' port='-1' autoport='yes'/>
```

4. この箇所に `passwd=パスワード` の形式で属性を追加して、ファイルを保存したあとエディタを終了します。なお、パスワードは最大 8 文字まで設定することができます。

```
<graphics type='vnc' port='-1' autoport='yes' passwd='PASSWORD'/>
```

5. `libvirtd` を再起動します:

```
> sudo systemctl restart libvirtd
```

6. 設定を変更する前から VM ゲスト を動作させていた場合は、それら全てを再起動します。再起動を行わない場合、VNC ではパスワード認証を使用することができません。



警告: VNC プロトコルのセキュリティについて

VNC プロトコルは安全なプロトコルであるとは考えられていません。パスワードは暗号化されて送信されるものの、暗号鍵と暗号化されたパスワードの両方を傍受することができてしまえば、解読できてしまう危険性があるためです。そのため、TLS/SSL で別途暗号化を行って VNC 接続するか、SSH の暗号化トンネルを介して接続することをお勧めします。`virt-viewer` や仮想マシンマネージャ, Remmina (詳しくは『リファレンス』、第4章「VNC によるリモートグラフィカルセッション」、4.2項「Remmina: リモートデスクトップクライアント」をお読みください) では、いずれの方式にも対応しています。

11.2 VM ホストサーバ への接続

`libvirt` を利用してハイパーバイザに接続するには、まず Uniform Resource Identifier (URI) を指定する必要があります。この URI は `virsh` や `virt-viewer` (VM ホストサーバ 内で `root` から接続する場合を除きます) で必要となるほか、仮想マシンマネージャ でも任意で指定することができます。仮想マシンマネージャ では URI ではなく、接続パラメータ (例: `virt-manager -c qemu:///system`) と呼ばれることもあるほか、URI をグラフィカルなインターフェイスで作成することもできます。詳しくは 11.2.2項「仮想マシンマネージャ による接続の管理」をお読みください。

```
ハイパーバイザ❶+プロトコル❷://ユーザ名@ホスト名❸/接続の種類❹
```

❶ ハイパーバイザを指定します。openSUSE Leap では `test` (テスト用), `qemu` (KVM), `xen` (Xen) のいずれかを指定します。このパラメータは必須です。

❷ リモートのホストに接続する場合、ここでプロトコルを指定します。`ssh` (SSH トンネル経由で接続), `tcp` (SASL/Kerberos 認証付きの TCP 接続), `tls` (x509 証明書による認証を行う TLS/SSL 接続) のいずれかを指定します。

❸ リモートのホストに接続する際のユーザ名とリモートホスト名を入力します。ユーザ名を指定しない場合、このコマンドを呼び出したユーザの名前 (`$USER`) を使用します。詳しくは下記をお読みください。TLS 接続の場合、ホスト名は x509 証明書内に書かれているものと厳密に一致している必要があります。

❹ `QEMU/KVM` ハイパーバイザに接続する場合、2 種類の接続方法があります。`system` を指定すると完全なアクセス権限を、`session` を指定すると制限されたアクセスを取得することができます。`session` アクセスは openSUSE Leap ではサポート対象外となりますので、このドキュメンテーションでは `system` のみを使用しています。

ハイパーバイザ接続 URI の例

`test:///default`
: ローカルでのテスト用ハイパーバイザへの接続を行います。

`qemu:///system` もしくは `xen:///system`
: ローカルホストにある QEMU ハイパーバイザや Xen ハイパーバイザに接続します。このとき、完全なアクセス権限 (system) を取得します。

`qemu+ssh://tux@mercury.example.com/system` もしくは `xen+ssh://tux@mercury.example.com/system`
: リモートのホスト mercury.example.com にある QEMU ハイパーバイザや Xen ハイパーバイザに接続します。このとき、接続は SSH トンネル経由で行います。

`qemu+tls://saturn.example.com/system` もしくは `xen+tls://saturn.example.com/system`
: リモートのホスト mercury.example.com にある QEMU ハイパーバイザや Xen ハイパーバイザに接続します。このとき、接続は TLS/SSL トンネル経由で行います。

さらに詳しい情報や例について、詳しくは https://libvirt.org/uri.html にある `libvirt` のドキュメンテーション (英語) をお読みください。



注記: URI 内でのユーザ名について

Unix ソケット認証を利用する場合、それがユーザ名／パスワード認証であっても、Polkit による認証であっても、ユーザ名は必ず指定しておく必要があります。これは全ての SSH 接続やローカル接続にも当てはまります。

SASL 認証 (TCP もしくは TLS 接続) を使用する場合や、TLS 接続で追加のサーバ側認証を行わない場合は、ユーザ名を指定する必要はありません。SASL の場合、ユーザ名は処理されません。ここでユーザ名を指定していても、SASL のユーザ名とパスワードの入力を求めるプロンプトが別途表示されます。

11.2.1 非特権ユーザに対する「システム」アクセス

上述のとおり、QEMU ハイパーバイザへの接続は 2 種類のプロトコル (`session` と `system`) を利用することができます。「session」を指定した場合、クライアント側のプログラムと同じ権限で動作することになります。この種類の接続は、さまざまな機能 (たとえば USB/PCI のデバイス割り当てや仮想ネットワークの設定、`libvirtd` へのリモートアクセスなど) が制限されていることから、デスクトップ仮想化向けに用意されている仕組みです。

「system」の接続はサーバの仮想化のために用意されている接続で、機能面の制限が無い代わりに、既定では `root` のみに対してアクセスを許可しています。しかしながら、`libvirt` に対する DAC (Discretionary Access Controll; 任意アクセス制御) ドライバの追加により、非特権ユーザに対しても「system」接続を許可することができるようになっています。たとえば `tux` に対して「system」接続を許可するには、下記のようにします:

手順 11.3: 一般ユーザに対する「システム」アクセスの許可

1. Unix ソケット経由でのアクセス許可を、11.1.1.1項「パーミッションとグループの所有権を利用した Unix ソケット向けのアクセス制御」に書かれている手順で有効化します。左記の例では、libvirt を `libvirt` というグループの全てのメンバーに対して許可し、このグループのメンバーに `tux` を追加しています。これにより、`tux` は `virsh` や 仮想マシンマネージャ を利用して接続することができるようになっています。

2. `/etc/libvirt/qemu.conf` ファイルを編集して、下記の設定を変更します:

   ```
   user = "tux"
   group = "libvirt"
   dynamic_ownership = 1
   ```

   これにより、VM ゲスト を `tux` から開始することができるようになるとともに、ゲストに対するリソース (仮想ディスクなど) の割り当てについても、`tux` からアクセスおよび変更できるようになります。

3. `tux` をグループ `kvm` のメンバーに追加します:

   ```
   > sudo usermod --append --groups kvm tux
   ```

   この手順は VM ゲスト を開始する際に使用する `/dev/kvm` へのアクセスを許可するために必要となるものです。

4. `libvirtd` を再起動します:

   ```
   > sudo systemctl restart libvirtd
   ```

11.2.2 仮想マシンマネージャ による接続の管理

仮想マシンマネージャ では管理対象となるそれぞれの VM ホストサーバ に対して `接続` を設定します。この接続には、対応するホスト内にある全ての VM ゲスト が含まれています。既定では、ローカルホストへの接続は設定済みで、接続も行われた状態になります。

全ての設定済みの接続は 仮想マシンマネージャ のメインウインドウ内に表示されます。接続済みのものには小さな三角形が描かれ、これを押すことで接続先の VM ゲスト の一覧を展開したり、折りたたんだりすることができます。

接続されていないものは灰色で描かれ、`未接続` と表示されます。ダブルクリックを行うか、マウスの右ボタンを押して、表示されたコンテキストメニューから［接続］を選択すると、接続を行うことができます。また、同じコンテキストメニューから［削除］を押すことで、接続を削除することもできます。



注記: 既存の接続の編集について

既存の接続を編集することはできません。接続を変更したい場合は、必要なパラメータで新しい接続を作成したあと、古いほうを削除してください。

仮想マシンマネージャ 内に新しい接続を追加するには、下記の手順を実施します:

1. ［ファイル］›［接続を追加］を選択します。
2. ホストの［ハイパーバイザー］(［Xen］もしくは［QEMU/KVM］) を選択します。
3. リモート接続を設定する場合は、［リモートホストに接続］を選択します。詳しくは 11.3項「リモート接続の設定」をお読みください。
   リモート接続の場合、`ユーザ名@ホスト名` の形式で、［ホスト名］に接続先を入力します。

   #### 重要: ユーザ名の指定

   TCP や TLS の接続の場合は、ユーザ名を指定する必要はありません。これらのプロトコルの場合は、ユーザ名は無視されます。しかしながら SSH 接続の場合、`root` 以外のユーザで接続する際には、ユーザ名の指定が必要となります。
4. 仮想マシンマネージャ の起動時に自動的に接続したくない場合は、［自動接続］のチェックを外してください。
5. 最後に［接続］を押すと、設定を保存することができます。

11.3 リモート接続の設定

`libvirt` の大きな利点の 1 つとして、1 箇所から複数のホスト内にある VM ゲスト を管理することができる、という点があります。本章では、リモート接続を行う際のサーバ側およびクライアント側の設定方法について説明しています。

11.3.1 SSH によるリモートトンネル (qemu+ssh もしくは xen+ssh)

VM ホストサーバ に対して SSH 経由でリモート接続を行う場合、必要な接続は SSH 接続だけとなります。それぞれ SSH デーモンが動作しているかどうか (`systemctl status sshd`) と、ファイアウォールで `SSH` のポートが開いていることを確認しておいてください。

SSH 接続の場合のユーザ認証は、従来型のファイルのユーザおよびグループの所有権、およびパーミッションでの制御 (11.1.1.1項「パーミッションとグループの所有権を利用した Unix ソケット向けのアクセス制御」で説明しています) で行います。ユーザ tux (`qemu+ssh://tux@mercury.example.com/system` もしくは `xen+ssh://tux@mercury.example.com/system`) での接続は何も設定することなく行うことができますので、`libvirt` 側で特に設定を行う必要はありません。

SSH 経由での接続 (`qemu+ssh://ユーザ名@システム` もしくは `xen+ssh://ユーザ名@システム`) を行う場合、`ユーザ名` に対するパスワードを入力する必要があります。これは『セキュリティ強化ガイド』、第23章「OpenSSH によるネットワーク操作の機密保持」、23.6項「公開鍵認証」の手順に従って公開鍵認証を設定することで、回避することができます。また、接続元のマシンで `gnome-keyring` を使用すると、さらに便利になります。詳しくは『セキュリティ強化ガイド』、第23章「OpenSSH によるネットワーク操作の機密保持」、23.9項「gnome-keyring を利用した公開鍵ログインの自動化」をお読みください。

11.3.2 x509 証明書を利用したリモート TLS/SSL 接続 (qemu+tls もしくは xen+tls)

TLS/SSL による暗号化で TCP 接続を行い、x509 証明書を利用して認証を行う環境の構築は SSH よりずっと複雑になりますが、拡張性をずっと高めることができます。この方式は、管理者の人数がしばしば変化するような VM ホストサーバ を管理する必要がある場合にお使いください。

11.3.2.1 基本的な考え方

TLS (Transport Layer Security) では証明書を利用することで、2 台のコンピュータ間の通信を暗号化します。接続を開始する側のコンピュータが常に「クライアント」となり、「クライアント証明書」を利用して接続を行います。また、接続を受け付ける側のコンピュータが常に「サーバ」となり、こちらも「サーバ証明書」を利用することになります。このような方式により、VM ホストサーバ を中央のデスクトップから一括管理することができるようになっています。

どちらのコンピュータからも接続を行うような場合、双方がクライアント証明書とサーバ証明書の両方を持つ必要があります。これはたとえば、VM ゲスト を一方のホストから他方のホストに移行させるような場合に当てはまります。

また、それぞれの x509 証明書には、対応する機密鍵ファイルが存在しています。証明書ファイルと機密鍵ファイルは一対で使用します。また、正しく証明書が認識されるようにするため、証明機関 (CA) と呼ばれる機関で署名され、発行された証明書を使用することをお勧めします。また、クライアントの証明書もサーバの証明書も、同じ CA から発行されているものでなければなりません。

重要: ユーザ認証について

TLS/SSL によるリモート接続は、2 台のコンピュータ間で特定の方向で認証を行い、通信を確立する処理を行うためのものです。特定のユーザにのみアクセスを許可したい場合は、クライアント側で証明書にアクセスすることのできるユーザを制限してください。詳しくは 11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」をお読みください。

`libvirt` では SASL を利用してサーバ側でユーザ認証を行うこともできます。こちらについては 11.3.2.6項「TLS ソケット向けの SASL を利用した中央管理型ユーザ認証」をお読みください。

11.3.2.2 VM ホストサーバ の設定

VM ホストサーバ が接続を受け付ける側になります。そのため、VM ホストサーバ には サーバ 証明書をインストールする必要があります。CA の証明書もインストールする必要があります。証明書の準備を整えたら、`libvirt` 側で TLS サポートを有効化することができます。

1. サーバ証明書を作成し、対応する CA の証明書とともにエクスポートします。
2. VM ホストサーバ 側で下記のとおりディレクトリを作成します:

   ```
   > sudo mkdir -p /etc/pki/CA/ /etc/pki/libvirt/private/
   ```

   下記のようにして証明書をインストールします:

   ```
   > sudo /etc/pki/CA/cacert.pem
   > sudo /etc/pki/libvirt/servercert.pem
   > sudo /etc/pki/libvirt/private/serverkey.pem
   ```

   ### 重要: 証明書へのアクセス制限について

   11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」で説明している内容に従って、証明書へのアクセスを制限するようにしてください。

3. 対応するソケットに対して TLS のサポートを有効化し、`libvirtd` を再起動します:

   ```
   > sudo systemctl stop libvirtd.service
   ```

```
> sudo systemctl enable --now libvirtd-tls.socket
> sudo systemctl start libvirtd.service
```

4. 既定では、`libvirt` は TCP ポート 16514 で TLS 接続を受け付けます。ファイアウオール側でこのポートを開いておいてください。

重要: TLS を有効化して libvirtd を再起動する場合の注意事項

`libvirt` で TLS を有効化した場合は、必ずサーバの証明書も設定しておいてください。証明書が設定されていないと、`libvirtd` の再起動が失敗します。また、証明書を変更した場合も、`libvirtd` の再起動が必要となります。

11.3.2.3 クライアントの設定とテスト

クライアントとは、接続を行うのことを指します。そのため、クライアント 証明書をインストールする必要があります。また、CA の証明書についてもインストールを行う必要があります。

1. クライアント証明書を作成し、対応する CA の証明書とともにエクスポートします。
2. クライアント側では下記のディレクトリを作成します:

```
> sudo mkdir -p /etc/pki/CA/ /etc/pki/libvirt/private/
```

下記のようにして証明書をインストールします:

```
> sudo /etc/pki/CA/cacert.pem
> sudo /etc/pki/libvirt/clientcert.pem
> sudo /etc/pki/libvirt/private/clientkey.pem
```

重要: 証明書へのアクセス制限について

11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」で説明している内容に従って、証明書へのアクセスを制限するようにしてください。

3. 下記のコマンドを実行して、クライアントとサーバの設定をテストします。下記では、`mercury.example.com` の箇所に VM ホストサーバ の名前を指定してください。なお、サーバ側の証明書を作成した際に使用したものと同じ、完全修飾ドメイン名を指定する必要があります。

```
#QEMU/KVM
```

```
virsh -c qemu+tls://mercury.example.com/system list --all

#Xen
virsh -c xen+tls://mercury.example.com/system list --all
```

設定が正しく行われていれば、VM ホストサーバ 内の `libvirt` に登録されている、全ての VM ゲスト の一覧が表示されるはずです。

11.3.2.4 TLS/SSL 接続向けの VNC の有効化

現時点では、TLS を利用した VNC 接続の暗号化は、少数のツールでしかサポートしていません。`tightvnc` や `tigervnc` などの一般的な VNC ビューアは、TLS/SSL に対応していません。仮想マシンマネージャ と `virt-viewer` を除くと、TLS/SSL に対応しているビューアは `remmina` のみとなります (詳しくは『リファレンス』、第4章「VNC によるリモートグラフィカルセッション」、4.2項「Remmina: リモートデスクトップクライアント」をお読みください) 。

11.3.2.4.1 VNC over TLS/SSL: VM ホストサーバ の設定

TLS/SSL を利用して VNC 経由でグラフィカルコンソールにアクセスするには、VM ホストサーバ を下記のように設定する必要があります:

1. まずはファイアウオールの設定を行い、サービス `VNC` 向けのポートを開きます。
2. `/etc/pki/libvirt-vnc` ディレクトリを作成し、このディレクトリ内に証明書へのリンクを作成します:

   ```
   > sudo mkdir -p /etc/pki/libvirt-vnc && cd /etc/pki/libvirt-vnc
   > sudo ln -s /etc/pki/CA/cacert.pem ca-cert.pem
   > sudo ln -s /etc/pki/libvirt/servercert.pem server-cert.pem
   > sudo ln -s /etc/pki/libvirt/private/serverkey.pem server-key.pem
   ```
3. `/etc/libvirt/qemu.conf` ファイルを編集して、下記のパラメータを設定します:

   ```
   vnc_listen = "0.0.0.0"
           vnc_tls = 1
           vnc_tls_x509_verify = 1
   ```
4. 最後に `libvirtd` を再起動します:

   ```
   > sudo systemctl restart libvirtd
   ```



重要: VM ゲスト を再起動する必要がある件について

VNC の TLS 設定は、VM ゲスト の起動時にのみ設定することができます。そのため、設定変更を行う前から起動している VM ゲスト が存在した場合、それらを再起動する必要があります。

11.3.2.4.2 VNC over TLS/SSL: クライアント設定

クライアント側でやるべきことは、選択したクライアントで認識される場所に、x509 のクライアント証明書を配置するだけです。ただ残念なことに、仮想マシンマネージャ と `virt-viewer` は、それぞれ別々の場所に証明書を配置する仕組みになっています。仮想マシンマネージャ は全てのユーザに反映できるシステム全体の場所のほか、ユーザごとの個別の場所も用意されています。Remmina (詳しくは **『リファレンス』、第4章「VNC によるリモートグラフィカルセッション」、4.2項「Remmina: リモートデスクトップクライアント」** をお読みください) では、リモートの VNC セッションの接続を開始する際に証明書の場所を尋ねます。

`仮想マシンマネージャ (virt-manager)`

リモートのホストに接続するには、まず 仮想マシンマネージャ を 11.3.2.3項「クライアントの設定とテスト」 で説明している手順に従って設定する必要があります。VNC で接続できるようにするには、クライアントの証明書をそれぞれ下記の場所に配置します:

システム全体に適用する場合の場所

```
/etc/pki/CA/cacert.pem
/etc/pki/libvirt-vnc/clientcert.pem
/etc/pki/libvirt-vnc/private/clientkey.pem
```

ユーザ単体で使用する場合の場所

```
/etc/pki/CA/cacert.pem
~/.pki/libvirt-vnc/clientcert.pem
~/.pki/libvirt-vnc/private/clientkey.pem
```

`virt-viewer`

`virt-viewer` では、下記の場所にあるシステム全体の設定のみを読み込みます:

```
/etc/pki/CA/cacert.pem
/etc/pki/libvirt-vnc/clientcert.pem
/etc/pki/libvirt-vnc/private/clientkey.pem
```

重要: 証明書へのアクセス制限について

11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」で説明している内容に従って、証明書へのアクセスを制限するようにしてください。

11.3.2.5 アクセス制限 (セキュリティ面の考慮事項)

x509 証明書は 2 つのものから作られています。1 つは一般に公開する証明書そのもの、そしてそれに対応する機密鍵です。これらの両方を使用することで認証を行うことができる仕組みです。そのため、クライアント証明書と機密鍵にアクセスができてしまえば、あとは自由に VM ホストサーバ にアクセスできることになってしまいます。サーバ証明書の場合、証明書と機密鍵にアクセスできてしまえば、誰でも VM ホストサーバ になりすますことができてしまいます。このようなことが発生しないようにするため、少なくとも機密鍵については、できる限りアクセス制限を厳しく設定する必要があります。これを実現するのに最も簡単な方法は、アクセスパーミッションを利用してアクセスを制限する方法です。

サーバ証明書

サーバ証明書は QEMU のプロセスが読み込むことができる必要があります。openSUSE Leap の QEMU では、`libvirt` ツールが起動するプロセスは `root` が所有者となりますので、`root` が証明書を読み込むことができれば十分となります:

```
> chmod 700 /etc/pki/libvirt/private/
> chmod 600 /etc/pki/libvirt/private/serverkey.pem
```

`/etc/libvirt/qemu.conf` ファイルを設定して QEMU のプロセスに対する所有権を変更している場合は、これらのファイルの所有権についても、あわせて設定を変更してください。

システム全体のクライアント証明書

システム全体で使用する機密鍵ファイルへのアクセスを制限するには、読み込みアクセスを特定のグループにのみ許可し、グループ内のメンバーのみが機密鍵ファイルを読み込めるようにしてください。下記の例では `libvirt` グループを作成し、`clientkey.pem` ファイルとその親ディレクトリの所有グループを `libvirt` に設定しています。あとはパーミッションを設定して、所有者と所有グループのみにアクセス権を与えます。最後に `tux` ユーザを `libvirt` グループのメンバーとして追加すると、機密鍵ファイルにアクセスできるようになります。

```
CERTPATH="/etc/pki/libvirt/"
# libvirt グループの作成
groupadd libvirt
# 所有者を root, 所有グループを libvirt に変更
chown root.libvirt $CERTPATH/private $CERTPATH/clientkey.pem
# パーミッションの制限
chmod 750 $CERTPATH/private
```

```
chmod 640 $CERTPATH/private/clientkey.pem
# ユーザ tux をグループ libvirt に追加
usermod --append --groups libvirt tux
```

ユーザごとの証明書

VNC 経由で VM ゲストのグラフィカルコンソールにアクセスするために使用する、ユーザごとのクライアント証明書は、ユーザのホームディレクトリ以下の `~/.pki` 内に配置します。SSH 等とは異なり、これらの証明書を使用する VNC ビューアは、機密鍵ファイルのパーミッションを確認しません。そのため、ユーザごとの証明書は、単純に自分自身が読み込めるようにパーミッションを設定するだけで十分となります。

11.3.2.5.1 サーバ側からのアクセス制限

既定では、対応する証明書を設定しているクライアントであれば、VM ホストサーバ はどのクライアントからでも接続を受け付けます。このような理由から、SASL によるサーバ側での認証を追加で設定できるようになっています (詳しくは 11.1.1.3項「SASL を利用したユーザ名とパスワードによる認証」をお読みください) 。

上記の方法以外にも、DN (Distinguished Name; 識別名) のホワイトリストを設定して、アクセスを制限する方法もあります。この場合、対応する DN の証明書を持つクライアントのみが接続できることになります。

具体的には、`/etc/libvirt/libvirtd.conf` ファイル内の `tls_allowed_dn_list` に、許可する DN の一覧を記述してください。この一覧にはワイルドカードを設定することもできます。ただし、空のリストを指定してはなりません。この場合、全ての接続を拒否することになってしまいます。

```
tls_allowed_dn_list = [
   "C=US,L=Provo,O=SUSE Linux Products GmbH,OU=*,CN=venus.example.com,EMAIL=*",
   "C=DE,L=Nuremberg,O=SUSE Linux Products GmbH,OU=Documentation,CN=*"]
```

証明書の識別名 (DN) を取得するには、下記のコマンドを使用します:

```
> certtool -i --infile /etc/pki/libvirt/clientcert.pem | grep "Subject:"
```

設定を変更したあとは、`libvirtd` を再起動します:

```
> sudo systemctl restart libvirtd
```

11.3.2.6 TLS ソケット向けの SASL を利用した中央管理型ユーザ認証

TLS では、直接的なユーザ認証を行うことはできません。実現するとすれば、11.3.2.5項「アクセス制限 (セキュリティ面の考慮事項)」で説明している手順に従って、クライアント側で証明書に対する読み込みアクセス権 (パーミッション) を設定することができるくらいです。ただし、中央管理型でサーバベー

スのユーザ認証が必要となる場合、`libvirt` では TLS 上で SASL (Simple Authentication and Security Layer) を使用するようにして、ユーザ認証を行うことができます。設定方法に関する詳細は、11.1.1.3項「SASL を利用したユーザ名とパスワードによる認証」をお読みください。

11.3.2.7 トラブルシューティング

11.3.2.7.1 仮想マシンマネージャ/ `virsh` がサーバに接続できない

下記の順序で確認を行ってください:

ファイアウオール側の問題 (TCP ポート 16514 がサーバ側で開く必要がある) ではありませんか?
クライアント証明書 (証明書そのものと機密鍵) が、仮想マシンマネージャ/ `virsh` を起動したユーザから読み込むことができますか?
接続時にサーバ証明書と同じ完全修飾ドメイン名を指定していますか?
サーバ側で TLS が有効化されています (`listen_tls = 1`) か?
サーバ側で `libvirtd` を再起動していますか?

11.3.2.7.2 VNC の接続が失敗する

まずは 仮想マシンマネージャ を利用して、リモートのサーバに接続できることを確認します。問題なく接続ができていることを確認したら、仮想マシンが TLS サポート付きで開始されているかどうかを確認します。下記の例では、`sles` という名前の仮想マシンに対して確認を行っています。

```
> ps ax | grep qemu | grep "\-name sles" | awk -F" -vnc " '{ print FS $2 }'
```

上記のコマンドの出力に下記のような内容が含まれていない場合は、TLS サポート付きで起動されていないことになりますので、設定を行って仮想マシンの再起動を行ってください。

```
-vnc 0.0.0.0:0,tls,x509verify=/etc/pki/libvirt
```

12 高度なストレージ設定

改訂履歴

2024-06-27

本章では、VM ホストサーバ の観点におけるストレージ関連の高度なトピックを説明しています。

12.1 virtlockd を利用したディスクファイルやブロックデバイスのロック (施錠)

ブロックデバイスやディスクファイルをロック (施錠) することで、これらのリソースを他の VM ゲストから書き込まれることがないように保護することができます。これにより、同一の VM ゲスト が二重に起動されることを防ぐことができるほか、異なる仮想マシンに同じディスクが割り当てられたりすることがないようになります。これにより、仮想マシンのディスクイメージが、誤った設定によって破壊されてしまうのを防ぐことにもなります。

ロック処理は `virtlockd` と呼ばれるデーモンが取り扱います。`libvirtd` デーモンとは個別に動作する仕組みであることから、`libvirtd` がクラッシュしたり再起動したりしてしまったような場合でも、ロックを提供し続けることができます。さらに `virtlockd` 自身の更新にも対応していて、自分自身で再起動を実施できるようになっています。これにより、`virtlockd` を更新しても、VM ゲスト を再起動する必要がありません。なお、`virtlockd` は KVM, QEMU, Xen にそれぞれ対応しています。

12.1.1 ロックの有効化

仮想ディスクのロックは openSUSE Leap の既定では有効化されていません。有効化してシステムの起動時に自動的に開始されるようにするには、下記の手順を実施します:

1. `/etc/libvirt/qemu.conf` ファイルを編集して、下記を設定します:

```
lock_manager = "lockd"
```

2. あとは過去のコマンドを実行して、`virtlockd` デーモンを開始します:

```
> sudo systemctl start virtlockd
```

3. `libvirtd` デーモンを再起動します:

```
> sudo systemctl restart libvirtd
```

4. システムの起動時に `virtlockd` が自動的に開始されるように設定します:

```
> sudo systemctl enable virtlockd
```

12.1.2 ロックの設定

既定では、`virtlockd` は VM ゲスト に設定された全てのディスクを自動的にロックします。また、既定の設定では「直接」ロック領域を使用し、VM ゲストの <disk> に書かれたデバイスに結びつけられたファイルに対して直接ロックを取得します。たとえば VM ゲスト 内に下記のような設定が存在した場合、`/var/lib/libvirt/images/my-server/disk0.raw` ファイルに対して `flock(2)` を直接実行して、ロックを獲得します:

```
<disk type='file' device='disk'>
 <driver name='qemu' type='raw'/>
 <source file='/var/lib/libvirt/images/my-server/disk0.raw'/>
 <target dev='vda' bus='virtio'/>
</disk>
```

`virtlockd` の設定は、設定ファイルである `/etc/libvirt/qemu-lockd.conf` を編集することで変更することができます。ここにはさらに詳しい情報の書かれたコメント (英語) が含まれています。なお、設定を変更した後は、それを反映させるために `virtlockd` を再読み込みさせる必要があります:

```
> sudo systemctl reload virtlockd
```

12.1.2.1 間接ロック領域の有効化

`virtlockd` の既定の設定では、"直接" ロック領域を使用し、VM ゲストの <disk> に書かれたデバイスに結びつけられたファイルに対して直接ロックを取得します。

ディスクのファイルパスが全てのホストから直接アクセスできるものではない場合、`virtlockd` を設定して「間接」ロック領域を使用するように設定することができます。この場合、間接ロック領域ディレクトリ内に、ディスクファイルパスのハッシュファイルを作成します。このロックは、実際のディスクファイルパスの代用として使用され、保持されるようになります。間接ロック領域は、`fcntl()` ロックに対応していないファイルシステムを使用する場合にも便利です。間接ロック領域は、`file_lockspace_dir` 設定で指定します:

```
file_lockspace_dir = "ロック領域のディレクトリ"
```

12.1.2.2 LVM や iSCSI ボリュームでのロック有効化

複数のホストで共有されている LVM や iSCSI ボリューム内に仮想ディスクが存在する場合、それらをロックするには、パス (既定で使用される方法) ではなく UUID で行う必要があります。それに加えてロック領域のディレクトリは、全てのホストからアクセスすることのできる共有の領域内に配置する必要があります。 `virtlockd` の LVM/iSCSI におけるロックの設定は、下記のようになります:

```
lvm_lockspace_dir = "ロック領域のディレクトリ"
iscsi_lockspace_dir = "ロック領域のディレクトリ"
```

12.2 ゲストのブロックデバイスのオンラインサイズ変更

状況によっては、ゲストシステムで使用されるブロックデバイスのサイズを変更し、サイズを大きくするか小さくする必要に迫られることがあります。たとえば元々割り当てていたサイズでは不足しているような場合には、サイズを増やす必要があります。ゲストに割り当てていたディスクが 論理ボリューム 内に存在していれば、ゲストシステムを動作させた状態のままサイズを変更することができます。これはオフラインによるディスクサイズの変更 (20.3項「guestfs ツール」内で説明している `virt-resize` コマンドに関する説明をお読みください) とは異なり、サイズ変更時にもゲストを停止させる必要がありませんので、大きな利点になります。VM ゲスト のディスクサイズを変更するには、下記の手順を実施します:

手順 12.1: ゲストディスクのオンラインサイズ変更

1. まずはゲストシステム内で、現在のサイズを確認します (下記では `/dev/vda` にディスクが配置されているものとします) 。

   ```
   # fdisk -l /dev/vda
   Disk /dev/sda: 160.0 GB, 160041885696 bytes, 312581808 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   ```

2. ホスト側で、ゲストの `/dev/vda` を提供している論理ボリュームのサイズを変更します。ここではたとえば 200 GB に変更します。

   ```
   # lvresize -L 200G /dev/mapper/vg00-home
   Extending logical volume home to 200 GiB
   Logical volume home successfully resized
   ```

3. ホスト側で、ゲストの `/dev/mapper/vg00-home` ディスクに対応するブロックデバイスのサイズを変更します。なお、下記の `ドメイン_ID` の箇所には、`virsh list` で表示されるドメイン名 (仮想マシン名) を入力します。

   ```
   # virsh blockresize  --path /dev/vg00/home --size 200G ドメイン_ID
   ```

```
ブロックデバイス '/dev/vg00/home' の容量が変更されました
```

4. ゲスト側で、新しいディスクサイズが認識されていることを確認します。

```
# fdisk -l /dev/vda
Disk /dev/sda: 200.0 GB, 200052357120 bytes, 390727260 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

12.3 ホストとゲストの間でのディレクトリ共有 (ファイルシステムのパススルー)

libvirt では、QEMU のファイルシステムパススルー (VirtFS とも呼ばれます) 機能を利用して、ホストとゲストの間でのディレクトリ共有を行うことができます。このようなディレクトリは複数の VM ゲストから同時にアクセスすることができるため、VM ゲスト 間でのファイル交換を行うこともできます。



注記: Windows ゲストとファイルシステムのパススルーについて

Windows ゲストには共有ディレクトリをマウントするためのデバイスドライバが提供されていないため、VM ホストサーバ と Windows ゲストとの間では、ファイルシステムのパススルー機能を利用してディレクトリの共有を行うことはできません。

VM ゲスト に対してディレクトリを共有するには、下記の手順を実施します:

1. 仮想マシンマネージャ でゲストのコンソールを開いて、メニューから［表示］›［詳細］を選択するか、もしくはツールバー内の［仮想マシンの情報を表示］を選択します。あとは［ハードウェアを追加］›［ファイルシステム］を選択して、［ファイルシステム・パススルー］を表示します。
2. ［ドライバー］では、［Handle］もしくは［Path］のいずれかを選択します。既定値は［Path］です。［モード］ではセキュリティモデルを指定します。これは、ホスト内でのファイルパーミッションの設定方法を指定します。下記のいずれかを選択します:

   ［Passthrough］(既定値)
   : ファイルシステム内のファイルを、クライアントユーザの権限で直接作成します。これは NFSv3 を利用している場合によく似ています。

   ［Squash］
   : ［Passthrough］と同様ですが、`chown` などの特権操作の失敗については無視します。これは KVM が `root` で動作していない場合に必要となります。

［Mapped］

ファイルサーバ側の権限（ `qemu.qemu` ）でファイルを作成します。ユーザ権限とクライアント側の権限情報は、拡張属性内に保存します。このモデルは、ホストとゲストを完全に分離しておきたい場合にお勧めの設定です。

3. VM ホストサーバ 内のディレクトリを［ソースパス］に指定します。なお、共有されたディレクトリをマウントする際には、［ターゲットパス］に指定した名前を使用します。また、この文字列はタグとして使用するだけのものであり、VM ゲスト 内のパスを表すものではありません。

4. ［完了］を押して設定を適用します。VM ゲスト が動作中の場合は、設定を反映させるのにシャットダウンが必要となります (ゲストの再起動では不十分です) 。

5. VM ゲスト を起動します。共有されたディレクトリをマウントするには、下記のコマンドを実行します:

```
> sudo mount -t 9p -o trans=virtio,version=9p2000.L,rw タグ /マウントポイント
```

共有されたディレクトリを恒久的にマウントしたい場合は、下記のような内容を `/etc/fstab` に追加します:

```
タグ    /マウントポイント    9p  trans=virtio,version=9p2000.L,rw    0   0
```

12.4 `libvirt` を利用した RADOS ブロックデバイスの使用

RADOS Block Devices (RBD) はデータを Ceph クラスタに保存します。このブロックデバイスでは、スナップショットやレプリケーション、データの一貫性維持などを行うことができます。お使いの `libvirt` 管理下の VM ゲスト から RBD を使用したい場合は、他のブロックデバイスと同様の手順を実施してください。

13 仮想マシンマネージャ を利用した仮想マシンの設定

改訂履歴

2024-06-27

仮想マシンマネージャ の［詳細］ビューでは、VM ゲスト の全ての設定やハードウエア構成に関する詳しい情報を表示することができます。このビューを利用することでゲストの設定を変更することができるほか、仮想ハードウエアを追加したりすることもできます。このビューを表示するには、仮想マシンマネージャ のゲストのコンソールを開いて、メニューバーから［表示］›［詳細］を選択するか、ツールバー内の［仮想マシンの情報を表示］を選択します。

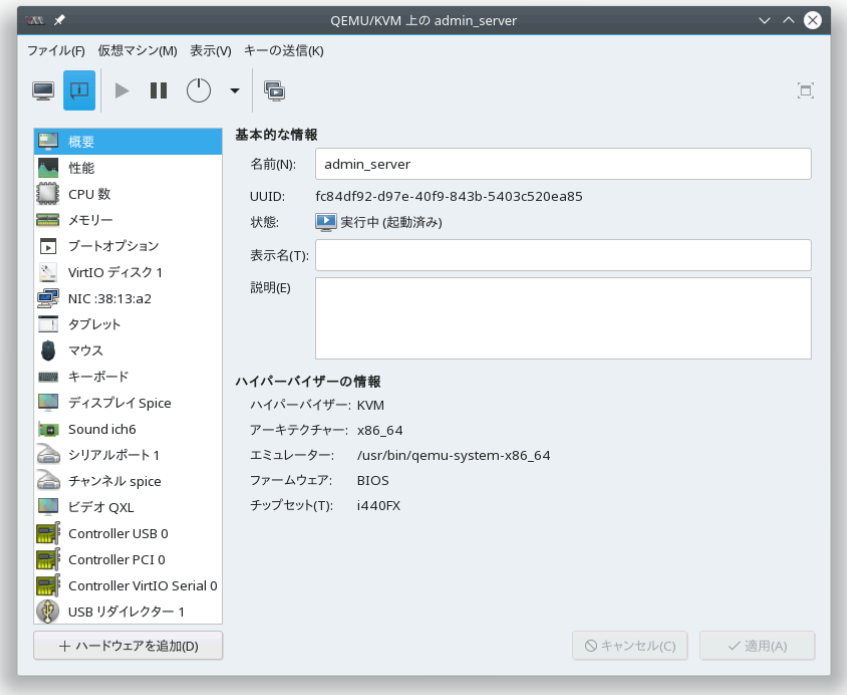


図 13.1: VM ゲスト の［詳細］ビュー

ウインドウの左側には VM ゲスト の概要と、インストール済みのハードウエアが一覧で表示されています。一覧内の項目を選択すると、右側の詳細ビューに詳細な設定が表示されます。必要なパラメータ変更を行ったあと、［適用］を押すことで設定を保存することができます。なお、一部の設定は即時に反映されますが、残りの多くはマシンの再起動が必要となります。再起動が必要な場合は、`virt-manager` がその旨を表すメッセージを表示します。

VM ゲスト からインストール済みのハードウエアを削除するには、左側のパネルで削除したいハードウエアを選択して、ウインドウの右下にある［削除］を押します。

新しいハードウエアを追加するには、左側のパネルの下にある［ハードウェアを追加］を押して、表示された［新しい仮想ハードウェアを追加］のウインドウ内で追加するハードウエアの種類を選択します。あとは必要なパラメータを設定して［完了］を押します。

下記の章には、それぞれの種類のハードウエアを 追加 する際の設定オプションについて説明しています。既存のハードウエアの設定変更は、追加と同じであるため、特に説明していません。

13.1 マシンの設定

本章では、仮想化されたプロセッサやメモリの設定について説明しています。これらのコンポーネントは VM ゲスト には必須のものであるため、削除はできません。このほか、概要や性能に関する情報の表示方法や、起動パラメータの変更方法についても説明しています。

13.1.1 概要

［概要］には、VM ゲスト とハイパーバイザに関する基本詳細情報が示されます。

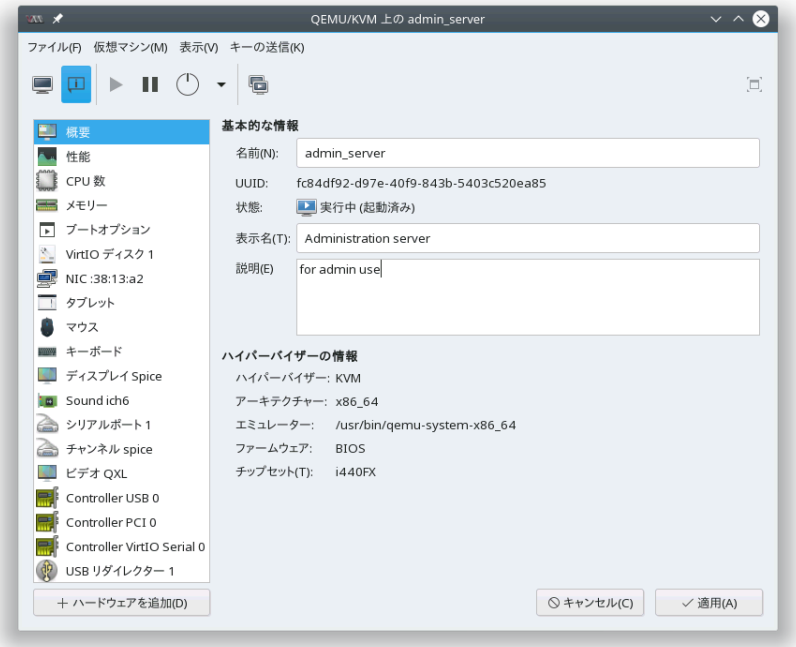


図 13.2: 概要の詳細

［名前］,［表示名］,［説明］はいずれも編集可能な項目で、［仮想マシンマネージャ］内で VM ゲスト を識別する際に使用するものです。



図 13.3: VM ゲスト の表示名と説明

［UUID］には仮想マシンを一位に識別するための値が示されます。また、［状態］には現在の状態（［実行中］，［一時停止中］，［シャットオフ］）が示されます。

［ハイパーバイザーの情報］セクションには、ハイパーバイザの種類と CPU アーキテクチャ、使用しているエミュレータとチップセットの種類が示されています。これらはいずれも変更することができません。

13.1.2 性能

［性能］では、CPU やメモリの使用率、ディスクやネットワークの I/O に関する自動更新型のグラフが表示されます。

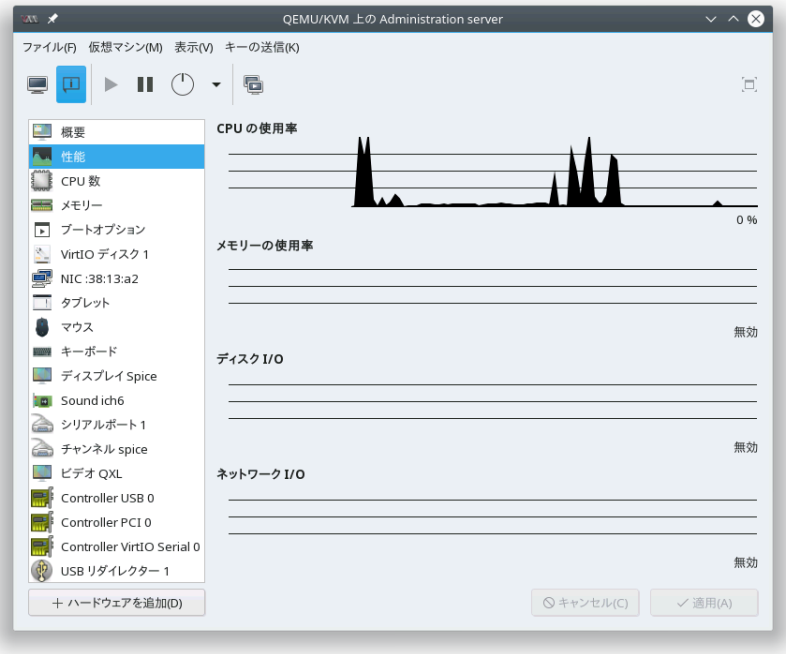


図 13.4: 性能

ヒント: 無効化されているグラフ表示の有効化について

既定では［性能］ビュー内のグラフは一部しか有効化されていません。これらのグラフも有効化したい場合は、［ファイル］›［仮想マシンマネージャーを表示］を選択したあと、［編集］›［設定］›［ポーリング］を選択して、定期的に更新したいグラフを選択してください。



図 13.5: 統計情報グラフ

13.1.3 CPU 数

[CPU 数] には、VM ゲスト のプロセッサ設定に関わる詳細情報が示されています。

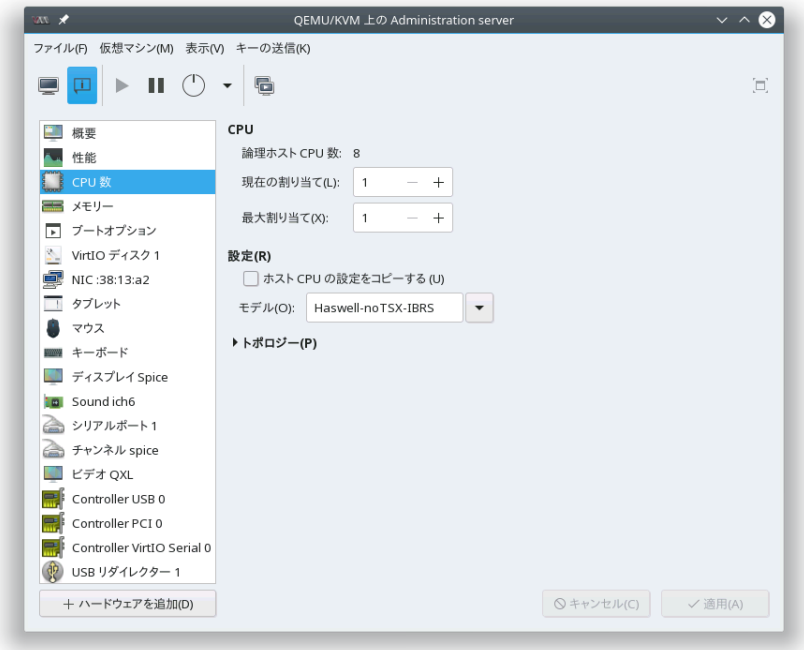


図 13.6: CPU 数のビュー

[CPU 数] のセクション内では、VM ゲスト の CPU の割当数に関連するいくつかのパラメータを設定することができます。また、[論理ホスト CPU 数] には、現時点で VM ホストサーバ に接続されていて、利用可能な CPU の数が示されています。

[設定] セクションでは、CPU のモデルやトポロジを設定することができます。

［ホスト CPU の設定をコピーする］を選択すると、VM ゲスト に対してホストの CPU モデルをそのまま公開するようになります。ホスト側の CPU モデルに関する詳細は、`virsh capabilities` コマンドの出力を確認してください。これを選択しない場合は、その下にあるドロップダウンボックスでモデルを選択することができます。

ホスト側の CPU モデルをそのまま VM ゲスト で使用すると、CPU の持つ機能を VM ゲスト 側でもそのまま使用できるようになる一方、VM ゲスト の移行時に問題が発生する可能性があることに注意する必要があります。また、`libvirt` では CPU の全ての機能を正確に擬似できているものではありませんので、VM ゲスト 側で使用できる CPU は VM ホストサーバ のものと全く同じにはならないことにも注意する必要があります。ただし、VM ゲスト に提供される ABI には再現性がありますので、CPU をきちんと指定した VM ゲスト であれば、移行元と移行先で厳密に同じ CPU モデルになるようになっています。

モデルとして `host-passthrough` を指定した場合は、VM ゲスト に提供される CPU は VM ホストサーバ の CPU と全く同一になります。これは、単純化された `host-model` CPU では提供されていない特別な機能を必要とする VM ゲスト を動作させる際には便利な設定です。ただし、この `host-passthrough` モデルの場合は、移行の柔軟性が低くなることに注意してください。具体的には、全く同一のハードウエア構成の VM ホストサーバ 間でのみ移行が可能となります。

`libvirt` の CPU モデルとトポロジのオプションについて、詳しくは https://libvirt.org/formatdomain.html#cpu-model-and-topology (英語) にある CPU model and topology のドキュメンテーションをお読みください。

［CPU トポロジーの手動設定］を選択すると、CPU のソケット数やコア数、スレッド数をそれぞれ独自に設定することができるようになります。

13.1.4 メモリ

［メモリー］には、VM ゲスト に対して提供するメモリに関わる詳細情報が示されます。

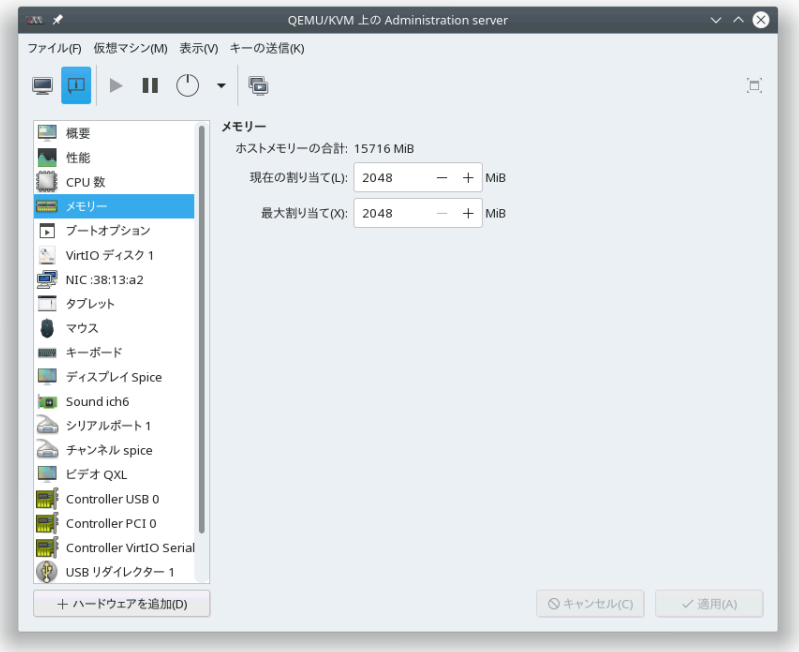


図 13.7: メモリのビュー

[ホストメモリーの合計]

VM ホストサーバ にインストールされているメモリ量の合計値を示しています。

[現在の割り当て]

現時点で VM ゲスト に対して提供しているメモリ量です。最大で［最大割り当て］の量まで、メモリをホットプラグで接続して増やすことができます。

[Enable shared memory]

仮想マシンに対して `memfd` バックエンドによる共有メモリの使用を許可するかどうかを指定します。これは virtiofs ファイルシステムを使用する場合に必要となります。詳しくは https://libvirt.org/kbase/virtiofs.html (英語) をお読みください。

[最大割り当て量]

現在のセッションで割り当てることのできるメモリ量の最大値です。この値を変更した場合、設定の反映は次回の VM ゲスト の再起動以降となります。

Enable launch security

VM ホストサーバ が AMD-SEV 技術に対応している場合、このオプションを有効化することで、メモリを暗号化してゲストを起動できるようになります。なお、このオプションを使用するには、仮想マシンに Q35 のチップセットを指定する必要があります。

重要: 巨大なメモリを使用する VM ゲスト の場合について

4TB 以上のメモリを必要とする VM ゲスト を動作させる場合、現時点では `host-passthrough` CPU モードを使用するか、`host-model` または `custom` CPU モードを使用する場合は仮想 CPU アドレスサイズを明示的に指定しなければなりません。これは、後者 2 つの CPU モードの場合、既定の仮想 CPU アドレスサイズでは 4 TB 以上のメモリに対して不適切な設定になってしまっているためです。なお、アドレスサイズの設定を行うには、VM ゲストの XML 設定ファイルを直接編集する必要があります。仮想 CPU アドレスサイズの設定について、詳しくは 14.6項「メモリ割り当ての設定」をお読みください。

13.1.5 ブートオプション

［ブートオプション］には、VM ゲスト の起動処理に関わるオプションが表示されます。

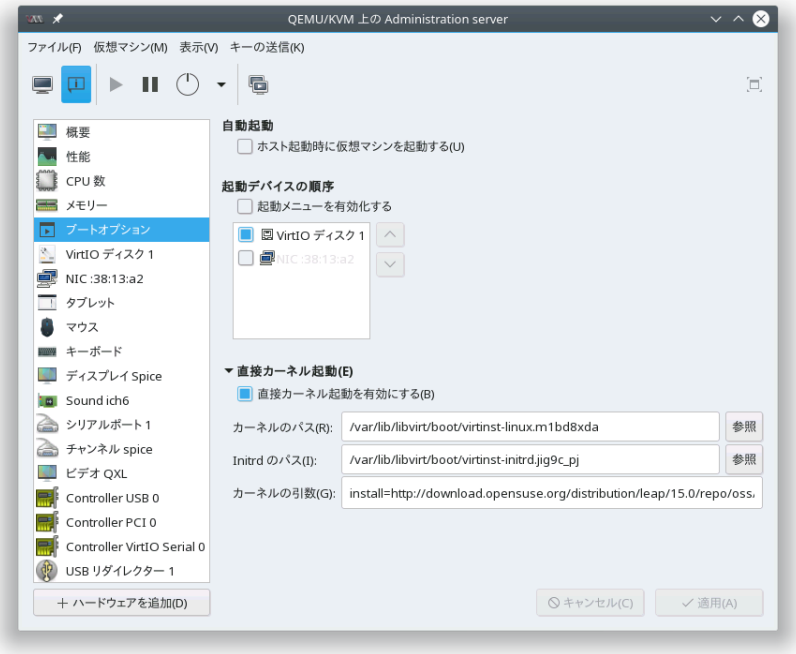


図 13.8: ブートオプション

［自動起動］セクションには、VM ホストサーバ の起動処理時に自動的に仮想マシンを起動するための設定が示されています。

［起動デバイスの順序］では、VM ゲスト を起動する際に使用するデバイスを選択します。また、一覧の右側にある上下の矢印ボタンを利用することで、使用順序を変更することもできます。なお、VM ゲスト の起動時にデバイスを選択するようにしたい場合は、［起動メニューを有効化する］を選択してください。

起動デバイス内に存在するものとは異なるカーネルを利用して起動したい場合は、［直接カーネル起動を有効にする］を選択して、VM ホストサーバ のファイルシステム内から、使用するカーネルと initrd のパスをそれぞれ選択してください。また、カーネルに対してパラメータを設定することもできます。

13.2 ストレージ

本章では、ストレージデバイスの設定に関する詳細な説明を行っています。ここでは、ハードディスクのほか、USB や CD-ROM ドライブなどのリムーバブルメディアについても取り扱っています。

手順 13.1: 新しいストレージデバイスの追加

1. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［ストレージ］を選択します。

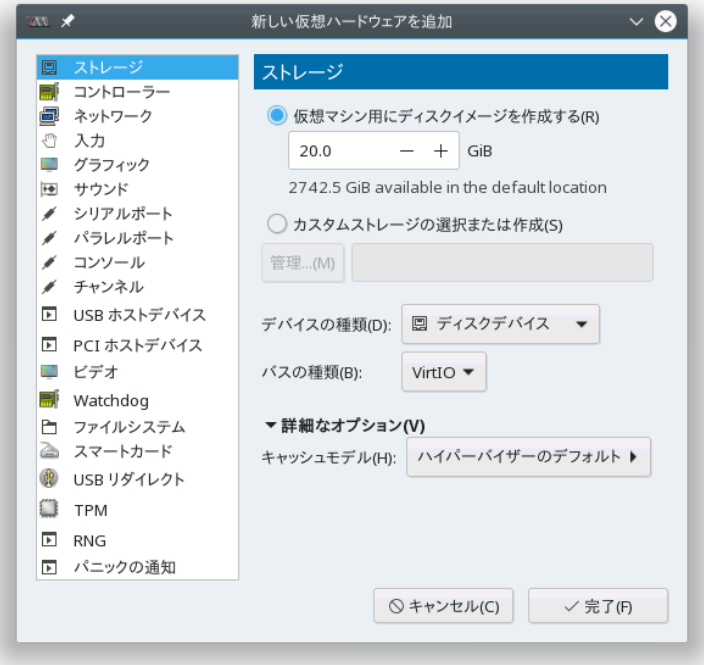


図 13.9: 新しいストレージの追加

2. 既定の場所に `qcow2` 形式のディスクイメージを作成するには、［仮想マシン用にディスクイメージを作成する］を選択して、サイズをギガバイト単位で指定します。

ディスクイメージの作成に際して、より細かい調整を行いたい場合は、［カスタムストレージの選択または作成］を選択し、［管理］を押してストレージプールとイメージの管理を行います。あとは［ストレージボリュームの選択］ウインドウで各種の設定を行います。設定項目に関する詳細は、8.2.2項「仮想マシンマネージャ を利用したストレージの管理」で説明している［ストレージ］とほぼ同じです。



ヒント: サポート対象のストレージ形式について

SUSE では、`raw` および `qcow2` のストレージ形式のみをサポートしています。

3. ディスクイメージファイルの作成および指定を行ったら、次は［デバイスの種類］を選択します。下記のいずれかを選択してください:

   - ［ディスクデバイス］
   - ［CDROM デバイス］: この場合、［仮想マシン用にディスクイメージを作成する］は選択できません。
   - ［フロッピーデバイス］: この場合、［仮想マシン用にディスクイメージを作成する］は選択できません。
   - ［LUN パススルー］: 既存の SCSI ストレージを、ストレージプールに追加せずに直接使用する必要があります。

4. 続いて、デバイスの［バスの種類］を選択します。利用可能な選択肢は、上記で選択したデバイスの種類によって異なります。［VirtIO］を選択すると、準仮想化ドライバを使用するようになります。

5. ［詳細なオプション］セクション内では、［キャッシュモデル］等を設定することができます。キャッシュ関連の設定について、詳しくは 第18章「ディスクのキャッシュモード」をお読みください。

6. 最後に［完了］を押して設定を保存します。新しいストレージデバイスが左側のパネルに表示されるようになります。

13.3 コントローラ

本章では、新しいコントローラの追加や設定について説明しています。

手順 13.2: 新しいコントローラの追加

1. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［コントローラー］を選択します。

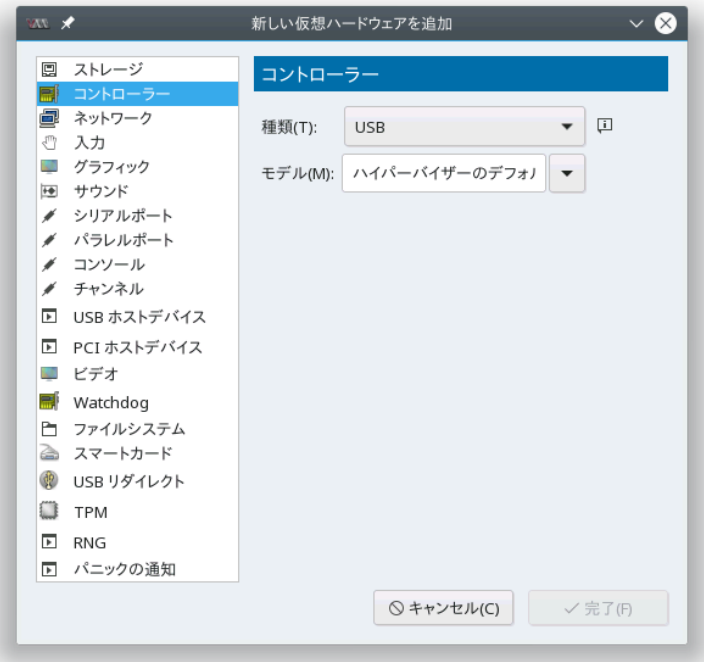


図 13.10: 新しいコントローラの追加

2. コントローラの種類を選択します。[IDE]，[Floppy]，[SCSI]，[SATA]，[VirtIO Serial] (準仮想化)，[USB]，[CCID] (スマートカードデバイス) の中から選択します。

3. USB や SCSI コントローラの場合、必要であればコントローラのモデルを選択します。

4. 最後に [完了] を押して設定を保存します。新しいコントローラが左側のパネルに表示されるようになります。

13.4 ネットワーク

本章では、新しいネットワークデバイスの追加や設定の方法を説明しています。

手順 13.3: 新しいネットワークデバイスの追加

1. 左側のパネルの下にある [ハードウェアを追加] を押して、[新しい仮想ハードウェアを追加] ウインドウを開きます。そのウインドウでは [ネットワーク] を選択します。



図 13.11: 新しいネットワークインターフェイスの追加

2. ［ネットワークソース］の一覧から、ネットワーク接続の接続先を指定します。この一覧には VM ホストサーバ 側で利用することのできる物理ネットワークインターフェイスのほか、ネットワークブリッジやネットワークボンディングなどが表示されます。また、既に設定済みの仮想ネットワークに接続することもできます。仮想マシンマネージャ による仮想ネットワークの設定方法について、詳しくは 8.1項「ネットワークの設定」をお読みください。

3. 次に、ネットワークデバイスに設定する［MAC アドレス］を指定します。仮想マシンマネージャ側では、乱数を元にした値を入力してありますが、お使いのネットワーク環境で問題があるような場合は、必要に応じて変更を行ってください。

4. さらに一覧からデバイスのモデルを選択します。［ハイパーバイザーのデフォルト］のほか、［e1000］,［rtl8139］,［virtio］のいずれかを指定することができます。なお、［virtio］を指定した場合は、準仮想化ドライバを使用することになります。

5. 最後に［完了］を押して設定を保存します。新しいネットワークデバイスが左側のパネルに表示されるようになります。

13.5 入力デバイス

本章では、マウスやキーボード、タブレットなどの新しい入力デバイスの追加や設定について説明しています。

手順 13.4: 新しい入力デバイスの追加

1. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［入力］を選択します。

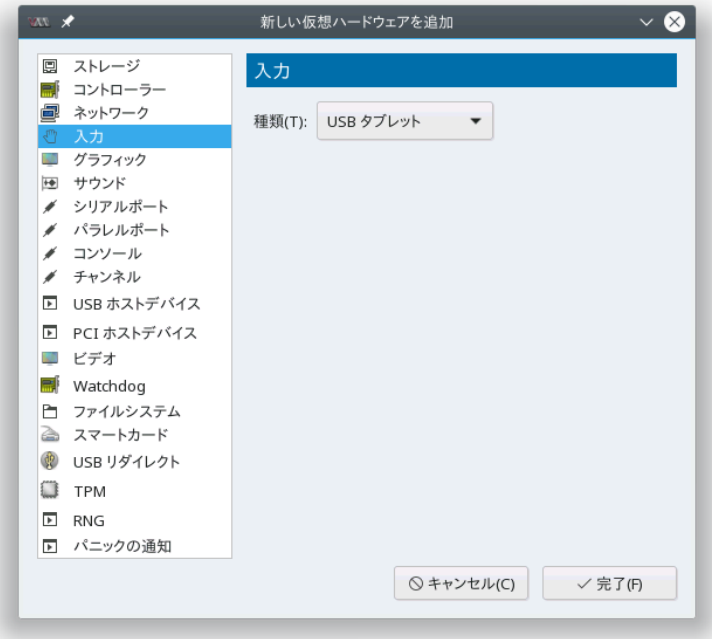


図 13.12: 新しい入力デバイスの追加

2. 一覧からデバイスの種類を選択します。

3. 最後に［完了］を押して設定を保存します。新しい入力デバイスが左側のパネルに表示されるようになります。

ヒント: マウスカーソルの捕捉について

VM ゲスト のコンソール内でマウスのボタンを押してしまうと、マウスカーソルがそのコンソールウインドウ内に捕捉されてしまい、コンソールの外側には移動できなくなってしまいます。このような場合は、 `Alt` – `Ctrl` を押して、明示的に解放を行ってください。また、マウスカーソルがコ

ンソール内に捕捉されず、VM ゲスト のウインドウの内側と外側を自由に行き来できるようにしたい場合は、手順13.4「新しい入力デバイスの追加」の手順に従って、［EvTouch USB グラフィックタブレット］を VM ゲスト に追加してください。

タブレットを追加することで、VM ゲスト を GUI で使用している場合、VM ホストサーバ と VM ゲスト の間をマウスカーソルが自由に行き来できるようになります。タブレットを追加しないと、上記のように捕捉状態になった場合にのみ、VM ゲスト 側のマウスカーソルを移動できるようになります。

13.6 ビデオ

本章では、新しいビデオデバイスの追加や設定の方法を説明しています。

手順 13.5: ビデオデバイスの追加

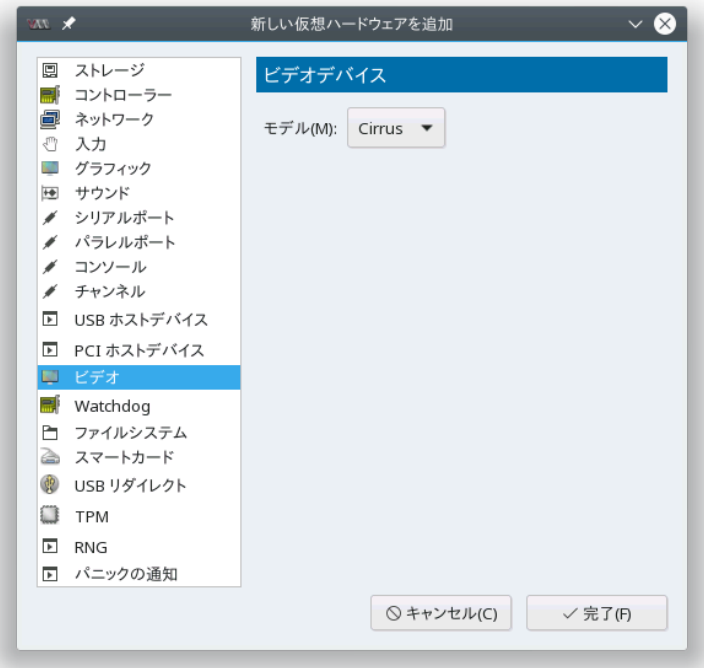
1. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［ビデオ］を選択します。
2. 

図 13.13: 新しいビデオデバイスの追加

3. ドロップダウンボックスから型式を選択します。

注記: セカンダリビデオデバイスについて

セカンダリビデオデバイスには、［QXL］と［Virtio］のみを追加することができます。

4. 最後に［完了］を押して設定を保存します。新しいビデオデバイスが左側のパネルに表示されるようになります。

13.7 USB リダイレクト

クライアントマシンに接続されている USB デバイスを VM ゲスト にそのまま転送したい場合は、［USB リダイレクト］を使用します。

手順 13.6: USB リダイレクトの追加

1. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［USB リダイレクト］を選択します。

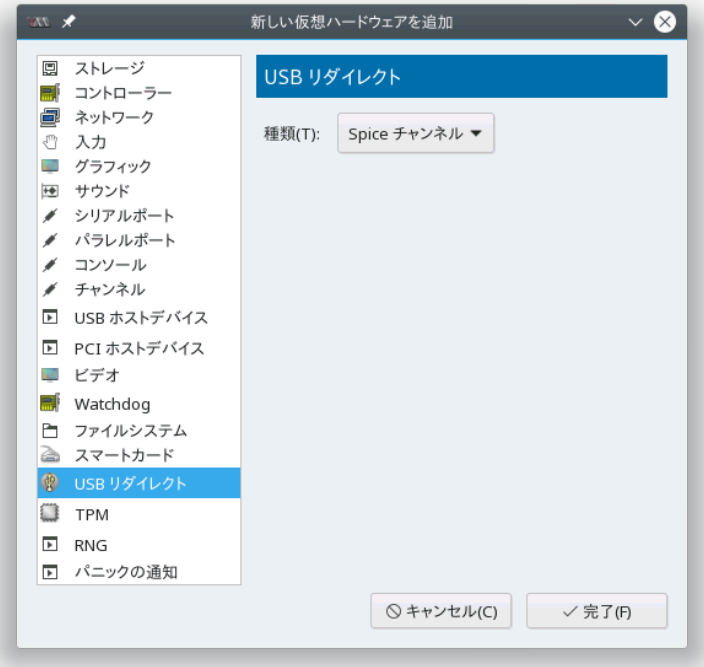


図 13.14: 新しい USB リダイレクトの追加

2. 一覧からデバイスの種類を選択します。ご利用の環境に合わせて［Spice チャンネル］もしくは［TCP］リダイレクタのいずれかを選択することができます。

3. 最後に［完了］を押して設定を保存します。新しい USB リダイレクトが左側のパネルに表示されるようになります。

13.8 その他

スマートカード

スマートカードの機能は［スマートカード］を選択することで追加することができます。物理的な USB スマートカードリーダを VM ゲスト に受け渡す動作になります。

ウオッチドッグ

仮想的なウオッチドッグデバイスにも対応しています。こちらは［Watchdog］を選択することで追加することができます。それぞれモデルとアクションを選択することができます。



ヒント: 仮想ウオッチドッグデバイスの要件について

QA 仮想ウオッチドッグデバイスでは、VM ゲスト 内に特定のドライバとデーモンをインストールする必要があります。インストールを行わないと、ウオッチドッグデバイスが動作しません。

TPM

ホスト側の TPM デバイスを VM ゲスト 側からアクセスできるようにするには、［TPM］を選択します。



ヒント: 仮想 TPM について

ホスト側の TPM は、同時に 1 つの VM ゲスト からしか使用することができません。

13.9 仮想マシンマネージャ を利用した CD/DVD-ROM デバイスの追加

KVM では、VM ゲスト に対して CD もしくは DVD-ROM デバイスを提供する機能に対応しています。VM ホストサーバ 側では VM ゲスト から物理的なドライブをそのままアクセスできるようにする設定のほか、ISO イメージファイル経由でアクセスする設定も行うことができます。既存の CD や DVD から ISO イメージを作成するには、`dd` コマンドを使用します:

```
> sudo dd if=/dev/CD_または_DVD_デバイス of=my_distro.iso bs=2048
```

VM ゲスト に対して CD/DVD-ROM デバイスを追加するには、下記の手順を実施します:

1. 仮想マシンマネージャ を起動し、設定を行いたい VM ゲスト の項目を選んでダブルクリックを行い、コンソールを開きます。その後、［表示］›［詳細］を選び、［詳細］ビューに切り替えます。
2. ［ハードウェアを追加］を押し、表示されたウインドウ内で［ストレージ］を選択します。
3. ［デバイスの種類］では［CD-ROM デバイス］を選択します。
4. ［カスタムストレージの選択または作成］を選択します。
    a. VM ホストサーバ 側の物理ドライブを割り当てる場合は、CD/DVD-ROM デバイスのパス (例: `/dev/cdrom`) を［管理］の右側に入力します。それ以外にも、［管理］を押して［ローカルを参照］を押し、デバイスを選択してもかまいません。ただし、物理ドライブをデバイスとして追加する処理は、仮想マシンマネージャ を VM ホストサーバ 内で実行した場合にのみ実施することができます。
    b. 既存のイメージを割り当てる場合は、［管理］を押してから、表示されたストレージプール内にあるイメージを選択してください。仮想マシンマネージャ を VM ホストサーバ 内で実行している場合は、［ローカルを参照］を押して、ファイルシステム内から選択することもできます。イメージを選択したら、［ボリュームの選択］を押します。
5. 最後に［完了］を押して設定を保存します。
6. VM ゲスト を再起動して、新しいデバイスにアクセスできるようにします。詳しくは 13.11項「仮想マシンマネージャ を利用したフロッピィもしくは CD/DVD-ROM メディアの取り出しと交換」をお読みください。

13.10 仮想マシンマネージャ を利用したフロッピィデバイスの追加

現時点では、KVM はフロッピィディスクイメージの使用のみに対応しています。物理的なフロッピィディスクドライブへのアクセスには対応していません。既存のフロッピィディスクメディアからイメージを作成するには、`dd` を使用します:

```
> sudo dd if=/dev/fd0 of=/var/lib/libvirt/images/floppy.img
```

空のフロッピィディスクイメージを作成する場合は、下記のいずれかのコマンドを実行します:

空のイメージ

```
> sudo dd if=/dev/zero of=/var/lib/libvirt/images/floppy.img bs=512 count=2880
```

FAT 形式でフォーマットされたイメージ

```
> sudo mkfs.msdos -C /var/lib/libvirt/images/floppy.img 1440
```

VM ゲスト に対してフロッピィディスクデバイスを追加するには、下記の手順を実施します:

1. 仮想マシンマネージャ を起動し、設定を行いたい VM ゲスト の項目を選んでダブルクリックを行い、コンソールを開きます。その後、［表示］› ［詳細］を選び、［詳細］ビューに切り替えます。
2. ［ハードウェアを追加］を押し、表示されたウインドウ内で［ストレージ］を選択します。
3. ［デバイスの種類］では［フロッピーデバイス］を選択します。
4. ［カスタムストレージの選択または作成］を選択して、［管理］を押して、ストレージプール内から既存のイメージを選択します。仮想マシンマネージャ を VM ホストサーバ 内で実行している場合は、［ローカルを参照］を押して、ファイルシステム内から選択することもできます。イメージを選択したら、［ボリュームの選択］を押します。
5. 最後に［完了］を押して設定を保存します。
6. VM ゲスト を再起動して、新しいデバイスにアクセスできるようにします。詳しくは 13.11項「仮想マシンマネージャ を利用したフロッピィもしくは CD/DVD-ROM メディアの取り出しと交換」をお読みください。

13.11　仮想マシンマネージャ を利用したフロッピィもしくは CD/DVD-ROM メディアの取り出しと交換

VM ホストサーバ に接続された物理的な CD/DVD-ROM デバイスを使用しているのか、それともISO やフロッピィイメージを使用しているのかによって、作業内容が異なります。いずれの場合であっても、VM ゲスト 内の既存のデバイス内にあるメディアを交換したり、イメージを交換したりするには、まずゲスト側で 切断 を行う必要があります。

1. 仮想マシンマネージャ を起動し、設定を行いたい VM ゲスト の項目を選んでダブルクリックを行い、コンソールを開きます。その後、［表示］› ［詳細］を選び、［詳細］ビューに切り替えます。

2. フロッピィディスクデバイスや CD/DVD-ROM デバイスを選択して［切断］を押し、メディアの「取り出し」を行います。

3. 新しいメディアを「挿入」するには、［接続］を押します。

    a. VM ホストサーバ の物理的な CD/DVD-ROM デバイスを使用している場合は、まずはデバイス内のメディアを交換します (なお、取り出しを行う前に VM ホストサーバ 側でマウント解除を行う必要があるかもしれません) 。その後、［Physical Device］を選択して、ドロップダウンボックスでデバイスを選択します。

    b. ISO イメージを使用している場合は、［Image Location］を選択してから、［参照］を押してイメージを選択します。リモートのマシンから接続している場合、既存のストレージプール内のイメージのみを選択することができます。

4. ［OK］を押して完了します。これで VM ゲスト から新しいメディアにアクセスできるようになります。

13.12 VM ゲスト に対するホスト側の PCI デバイスの割り当て

ホスト側に接続された PCI デバイスをゲストに直接割り当てることができます (PCI パススルーと言います) 。いずれかの VM ゲスト に PCI デバイスを割り当てると、解放を行わない限り、ホスト側からはアクセスができなくなるほか、他の VM ゲスト からもアクセスすることができなくなります。また、VM ホストサーバ でこの機能を使用するには、重要: VFIO および SR-IOV の要件について で説明している設定を行う必要があります。

13.12.1 仮想マシンマネージャ を利用した PCI デバイスの追加

下記の手順では、仮想マシンマネージャ を利用してホストマシンの PCI デバイスを VM ゲスト に割り当てる方法を説明しています:

1. 仮想マシンマネージャ を起動し、設定を行いたい VM ゲスト の項目を選んでダブルクリックを行い、コンソールを開きます。その後、［表示］› ［詳細］を選び、［詳細］ビューに切り替えます。

2. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［PCI ホストデバイス］を選択します。

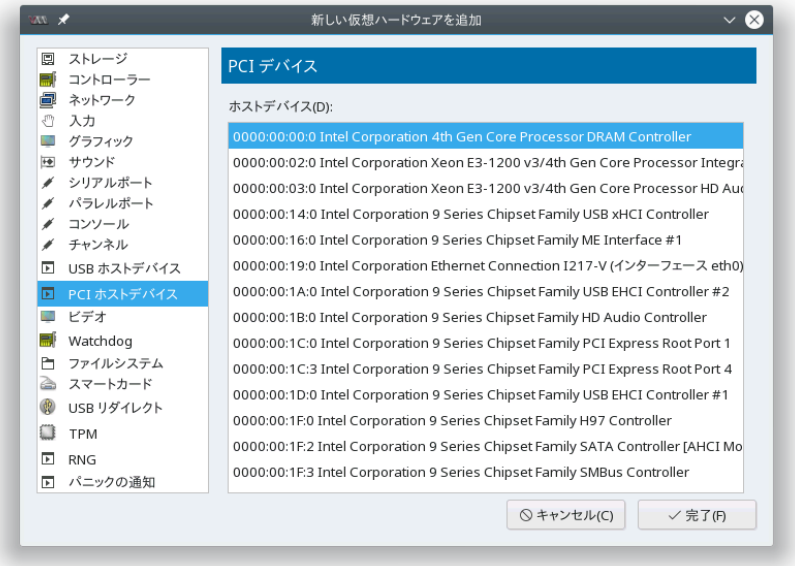


図 13.15: PCI デバイスの追加

3. 利用可能な PCI デバイスの一覧の中から、ゲストに追加したいデバイスを選択します。選択を行ったら［完了］を押します。

重要: SLES 11 SP4 の KVM ゲストについて

新しい QEMU マシンタイプ (pc-i440fx-2.0 もしくはそれ以降) を設定した SLES 11 SP4 KVM ゲストの場合、ゲスト内では既定で `acpiphp` モジュールが読み込まれません。このモジュールは、ディスクやネットワークデバイスのホットプラグ (活性接続) を行うために読み込んでおかなければならないモジュールですので、必要であれば `modprobe acpiphp` コマンドを実行して読み込んでください。なお、`/etc/modprobe.conf.local` ファイル内に `install acpiphp /bin/true` の行を追加すると、システムの起動時に自動読み込みを行うことができます。

重要: QEMU Q35 マシンタイプを使用する KVM ゲストについて

QEMU Q35 マシンタイプを使用する KVM マシンの場合、1 つの `pcie-root` コントローラと 7 つの `pcie-root-port` コントローラからなる PCI トポロジを構成します。`pcie-root` コントローラはホットプラグ (活性接続) には対応しませんが、`pcie-root-port` コントローラはそれぞれ 1 つの PCIe デバイスのホットプラグに対応します。PCI コントローラ自身はホットプラグに対応していませんので、7 つ以上の PCIe デバイスのホットプラグが必要となる場合、あらかじめ `pcie-root-port` コントローラを追加しておくようにしてください。また `pcie-`

`to-pci-bridge` コントローラを追加することで、古い PCI デバイスのホットプラグを実現することもできます。QEMU のマシンタイプ別の PCI トポロジの詳細について、詳しくは https://libvirt.org/pci-hotplug.html (英語) をお読みください。

13.13 VM ゲスト に対するホスト側の USB デバイスの割り当て

PCI デバイスの割り当て (13.12項「VM ゲスト に対するホスト側の PCI デバイスの割り当て」で説明しています) と同様に、ホスト側の USB デバイスをゲストに割り当てることもできます。いずれかの VM ゲスト に USB デバイスを割り当てると、解放を行わない限り、ホスト側からはアクセスができなくなるほか、他の VM ゲスト からもアクセスすることができなくなります。

13.13.1 仮想マシンマネージャ を利用した USB デバイスの追加

仮想マシンマネージャ を利用してホスト側の USB デバイスを VM ゲスト に割り当てるには、下記の手順を実施します:

1. 仮想マシンマネージャ を起動し、設定を行いたい VM ゲスト の項目を選んでダブルクリックを行い、コンソールを開きます。その後、［表示］› ［詳細］を選び、［詳細］ビューに切り替えます。
2. 左側のパネルの下にある［ハードウェアを追加］を押して、［新しい仮想ハードウェアを追加］ウインドウを開きます。そのウインドウでは［USB ホストデバイス］を選択します。



図 13.16: USB デバイスの追加

3. 利用可能な USB デバイスの一覧の中から、ゲストに追加したいデバイスを選択します。選択を行ったら［完了］を押します。新しい USB デバイスが［詳細］ビュー内の左側に表示されるようになります。



ヒント: USB デバイスの取り外しについて

USB デバイスの割り当てを削除するには、［詳細］ビュー内の左側で対象のデバイスを選択して、［削除］を押します。

14 `virsh` を利用した仮想マシンの設定

改訂履歴

2025-01-29

VM ゲストを設定するにあたっては、仮想マシンマネージャ だけでなく `virsh` も使用することができます。`virsh` は仮想マシン (VM) を管理するためのコマンドラインツールで、これを利用することで、VM の状態の制御や設定の編集、他のホストへの移行などを行うことができます。下記の章では、`virsh` を利用した管理の方法について説明しています。

14.1 VM の設定変更

仮想マシンの設定ファイルは `/etc/libvirt/qemu/` 内に XML 形式で保存されていて、下記のような内容になっているはずです:

例 14.1: XML 設定ファイルの例

```
<domain type='kvm'>
  <name>sles15</name>
  <uuid>ab953e2f-9d16-4955-bb43-1178230ee625</uuid>
  <memory unit='KiB'>2097152</memory>
  <currentMemory unit='KiB'>2097152</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-q35-2.0'>hvm</type>
  </os>
  <features>...</features>
  <cpu mode='custom' match='exact' check='partial'>
    <model fallback='allow'>Skylake-Client-IBRS</model>
  </cpu>
  <clock>...</clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>...</disk>
  </devices>
```

```
  ...
</domain>
```

VM ゲスト の設定を変更する場合は、まずシャットオフ状態になっているかどうかを確認します:

```
> sudo virsh list --inactive
```

上記のコマンドの実行結果に編集対象の VM ゲスト が現れている場合は、そのまま設定を変更してかまいません:

```
> sudo virsh edit VM_名
```

なお設定の保存時には、`virsh` が RelaxNG スキーマを利用して設定内容のチェックを行います。

14.2 マシンの種類の変更

`virt-install` ツールを利用してインストールを行った場合、VM ゲスト のマシンの種類は、既定で pc-q35 になります。マシンの種類も VM ゲスト の設定ファイル内に含まれていて、`type` というタグ要素内に書かれています:

```
<type arch='x86_64' machine='pc-q35-2.3'>hvm</type>
```

下記の手順では、例としてマシンの種類を `q35` に変更します。`q35` という値は Intel* 社のチップセットを表す文字列で、PCIe が含まれているほか、最大で 12 個までの USB ポートに対応し、SATA や IOMMU にも対応しています。

手順 14.1: マシンの種類の変更

1. まずは VM ゲスト が停止していることを確認します:

   ```
   > sudo virsh list --inactive
   Id    名前                             状態
   ----------------------------------------------------
   -     sles15                           シャットオフ
   ```

2. この VM ゲスト に対する設定を編集します:

   ```
   > sudo virsh edit sles15
   ```

3. タグ内の `machine` という属性の値を、`pc-q35-2.0` に変更します:

   ```
   <type arch='x86_64' machine='pc-q35-2.0'>hvm</type>
   ```

4. VM ゲストを起動し直します:

```
> sudo virsh start sles15
```

5. マシンの種類が変更されていることを確認します。VM ゲストを起動してログインし、下記のコマンドを実行します:

```
> sudo dmidecode | grep Product
Product Name: Standard PC (Q35 + ICH9, 2009)
```



ヒント: マシンの種類を更新した際の推奨事項について

ホストシステム側の QEMU のバージョンをアップグレードした場合 (たとえば VM ホストサーバのディストリビューションのバージョンをアップグレードした場合など) 、VM ゲスト側のマシンの種類についても、利用可能な最新版にアップグレードするようにしてください。どのような種類を指定できるのかを知りたい場合は、VM ホストサーバ側で `qemu-system-x86_64 -M help` を実行してください。

また、既定で使用されるマシン種類 (`pc-i440fx`) なども、定期的に更新が行われます。お使いの VM ゲストが今も `pc-i440fx-1.X` のマシン種類で動作している場合は、`pc-i440fx-2.X` にアップグレードすることを強くお勧めします。これにより、最新の更新内容を受けることができることになり、不具合の修正や将来の互換性維持に役立つことになります。

14.3 ハイパーバイザ機能の設定

`libvirt` はほとんどの状況下において適切なハイパーバイザ設定を行うよう自動設定されていますが、必要に応じて特定の機能を有効化したり無効化したりすることができます。たとえば Xen の既定では PCI パススルーが有効化されていませんが、`passthrough` を設定することで有効化することができます。ハイパーバイザの機能は `virsh` を利用して、VM ゲストの設定ファイル内に `<features>` 要素を指定して設定します。たとえば Xen での PCI パススルー機能を有効化するには、下記のように設定します:

```
> sudo virsh edit sle15sp1
 <features>
    <xen>
      <passthrough/>
    </xen>
 </features>
```

あとは設定を保存して VM ゲストを再起動してください。

詳しくは https://libvirt.org/formatdomain.html#elementsFeatures (英語) にある libvirt の Domain XML format 内の Hypervisor features の章をお読みください。

14.4 起動オプションの変更

VM ゲスト の起動メニューの設定は `os` 要素内に含まれ、下記のような内容になっています:

```
<os>
  <type>hvm</type>
  <loader>readonly='yes' secure='no' type='rom'/>/usr/lib/xen/boot/hvmloader</loader>
  <nvram template='/usr/share/OVMF/OVMF_VARS.fd'/>/var/lib/libvirt/nvram/guest_VARS.fd</
nvram>
  <boot dev='hd'/>
  <boot dev='cdrom'/>
  <bootmenu enable='yes' timeout='3000'/>
  <smbios mode='sysinfo'/>
  <bios useserial='yes' rebootTimeout='0'/>
  </os>
```

上記の例では、`hd` と `cdrom` という 2 つのデバイスが有効化されています。設定の順序は実際の起動順序にも影響しますので、たとえば上記の例では `cdrom` よりも前に `hd` の起動が試されることになります。

14.4.1 起動順序の変更

VM ゲスト の起動順序は、XML 設定ファイル内での出現順序で表されます。つまり、デバイスのタグを入れ替えることで起動順序を変更できることになります。

1. VM ゲスト の XML 設定ファイルを開きます。

   ```
   > sudo virsh edit sles15
   ```

2. デバイスの順序を入れ替えます。

   ```
   ...
   <boot dev='cdrom'/>
   <boot dev='hd'/>
   ...
   ```

3. VM ゲスト の BIOS 設定内の起動メニューを確認して、起動順序が変更されていることを確認します。

14.4.2 直接カーネル起動の使用

直接カーネル起動を使用することで、ホスト内に保存されているカーネルと initrd を利用して起動を行うことができます。この場合は、`kernel` と `initrd` のタグを追加してファイルを指定します:

```
<os>
    ...
  <kernel>/root/f8-i386-vmlinuz</kernel>
  <initrd>/root/f8-i386-initrd</initrd>
    ...
<os>
```

直接カーネル起動を有効化するには、下記の手順を実施します:

1. VM ゲスト の XML 設定を開きます:

   ```
   > sudo virsh edit sles15
   ```

2. `os` タグ内に `kernel` タグを追加し、ホスト側でのカーネルファイルのパスを指定します:

   ```
   ...
   <kernel>/root/f8-i386-vmlinuz</kernel>
   ...
   ```

3. 同様に `initrd` タグを追加し、ホスト内での initrd ファイルのパスを指定します:

   ```
   ...
   <initrd>/root/f8-i386-initrd</initrd>
   ...
   ```

4. あとは VM を起動すると、新しいカーネルでの起動が行われます:

   ```
   > sudo virsh start sles15
   ```

14.5 CPU の設定

`virsh` を利用することで、VM ゲスト に対して提供される様々な仮想 CPU の設定を変更することができます。VM ゲスト に割り当てる CPU 数の現在値や最大値のほか、CPU のモデルや機能セットなども設定することができます。下記の章では、VM ゲスト で一般的な CPU 設定を変更する方法を説明しています。

14.5.1 CPU 数の設定

VM ゲスト の CPU の割当数は、`/etc/libvirt/qemu/` にある XML 設定ファイル内の `vcpu` タグ内に書かれています:

```
<vcpu placement='static'>1</vcpu>
```

上記の例では、VM ゲスト 側には CPU を 1 つだけ割り当てていることになります。下記の手順では、VM ゲスト 側への CPU の割当数の変更方法を説明しています:

1. まずは VM ゲスト が停止していることを確認します:

```
> sudo virsh list --inactive
Id    名前                             状態
----------------------------------------------------
-     sles15                           シャットオフ
```

2. 既存の VM ゲスト に対する設定を編集します:

```
> sudo virsh edit sles15
```

3. CPU の割当数を変更して保存します:

```
<vcpu placement='static'>2</vcpu>
```

4. VM ゲスト を起動し直します:

```
> sudo virsh start sles15
```

5. VM に対して割り当てられている CPU 数が変更されていることを確認します:

```
> sudo virsh vcpuinfo sled15
VCPU:           0
CPU:            N/A
State:          N/A
CPU time        N/A
CPU Affinity:   yy

VCPU:           1
CPU:            N/A
State:          N/A
CPU time        N/A
CPU Affinity:   yy
```

VM ゲスト が動作中の場合であっても、c CPU 数を変更することができます。ただし、VM ゲスト の起動時に設定した最大数までしかホットプラグ (活性接続) することができませんし、逆に減らす場合の最小数は 1 であることにも注意してください。下記の例では、現在の CPU 数である 2 を、あらかじめ設定しておいた最大値 4 に変更する手順を説明しています。

1. まずは現時点での CPU 数を確認します:

```
> sudo virsh vcpucount sles15 | grep live
maximum      live           4
current      live           2
```

2. CPU 数の現在値 (使用可能な CPU 数) を 4 に変更します:

```
> sudo virsh setvcpus sles15 --count 4 --live
```

3. 最後に vcpu 数が 4 になっていることを確認します:

```
> sudo virsh vcpucount sles15 | grep live
maximum      live           4
current      live           4
```

14.5.2 CPU モデルの設定

VM ゲスト に対して提示される CPU モデルの設定は、その中で動作する処理に影響します。既定の CPU モデルは `host-model` という設定値で、ホスト側の CPU モデルをそのまま VM ゲスト に提示する設定になります。

```
<cpu mode='host-model'/>
```

CPU モデルとして `host-model` を指定して VM ゲスト を起動すると、`libvirt` はホストの CPU モデルをそのままコピーして VM ゲスト 側に対して提供するようになります。VM ゲスト 側の定義にコピーされた CPU モデルと機能については、`virsh capabilities` コマンドの出力で確認することができます。

それ以外にも、`host-passthrough` という値を設定することもできます。

```
<cpu mode='host-passthrough'/>
```

CPU モデルとして `host-passthrough` を指定した場合は、VM ゲスト に提供される CPU は VM ホストサーバ と全く同一になります。これは、単純化された `host-model` CPU では提供されていない特別な機能を必要とする VM ゲスト を動作させる際には便利な設定です。ただし、この `host-passthrough` モデルの場合は、移行の柔軟性が低くなることに注意してください。具体的には、全く同一のハードウエア構成の VM ホストサーバ 間でのみ移行が可能となります。

このほか、`host-passthrough` の CPU モデルを指定した場合で、特定の不要な機能のみを無効化したいような場合は、下記のように設定することもできます。下記の例では、ホスト側の CPU と全く同一のモデルでありながら、`vmx` 機能のみを無効化しています。

```
<cpu mode='host-passthrough'>
  <feature policy='disable' name='vmx'/>
  </cpu>
```

また、`custom` CPU モデルを指定することで、CPU モデルを標準化し、クラスタ内にある異なる CPU 構成のホスト間の移行を可能にすることができます。たとえばクラスタ内に Nehalem, IvyBridge, SandyBridge の各 CPU モデルが混在しているような場合、VM ゲスト で `custom` CPU モデルを指定することで、VM ゲスト の CPU モデルを Nehalem に統一することができます。

```
<cpu mode='custom' match='exact'>
  <model fallback='allow'>Nehalem</model>
  <feature policy='require' name='vme'/>
  <feature policy='require' name='ds'/>
  <feature policy='require' name='acpi'/>
  <feature policy='require' name='ss'/>
  <feature policy='require' name='ht'/>
  <feature policy='require' name='tm'/>
  <feature policy='require' name='pbe'/>
  <feature policy='require' name='dtes64'/>
  <feature policy='require' name='monitor'/>
  <feature policy='require' name='ds_cpl'/>
  <feature policy='require' name='vmx'/>
  <feature policy='require' name='est'/>
  <feature policy='require' name='tm2'/>
  <feature policy='require' name='xtpr'/>
  <feature policy='require' name='pdcm'/>
  <feature policy='require' name='dca'/>
  <feature policy='require' name='rdtscp'/>
  <feature policy='require' name='invtsc'/>
  </cpu>
```

`libvirt` の CPU モデルとトポロジのオプションについて、詳しくは https://libvirt.org/formatdomain.html#cpu-model-and-topology (英語) にある CPU model and topology のドキュメンテーションをお読みください。

14.6 メモリ割り当ての設定

VM ゲスト に対するメモリ割当量の設定変更は、`virsh` でも行うことができます。メモリ割当量は `memory` 要素内に書かれていて、この中に VM ゲスト の起動時の最大メモリ割当量を設定します。また `currentMemory` という任意指定の要素を指定することで、VM ゲスト に割り当てる実

際のメモリ量を指定することもできます。なお、`currentMemory` の値を `memory` よりも少なくしておくことで、VM ゲスト の動作中にメモリ量を増やす (バルーン と呼びます) ことができます。また、`currentMemory` の指定を省略すると、`memory` と同じ値を指定したものと見なされます。

なお、VM ゲスト の設定を編集することで、メモリ設定を変更することができますが、変更した内容は次回の起動時まで反映されないことに注意してください。下記の手順では、VM ゲスト に対して起動時に 4GB のメモリを割り当て、あとから 8GB まで増やす場合の例を示しています:

1. VM ゲスト の XML 設定を開きます:

```
> sudo virsh edit sles15
```

2. `memory` タグを検索して、メモリの割当量を 8GB に変更します:

```
...
<memory unit='KiB'>8388608</memory>
...
```

3. `currentMemory` 要素が存在していない場合は `memory` 要素内に `currentMemory` 要素を追加します。既に存在している場合は、値のみを変更します:

```
[...]
<memory unit='KiB'>8388608</memory>
<currentMemory unit='KiB'>4194304</currentMemory>
[...]
```

VM ゲスト の動作中にメモリ割当量を変更したい場合は、`setmem` サブコマンドを使用します。下記の手順では、メモリの割り当てを 8GB まで増やしています:

1. 現時点での VM ゲスト のメモリ設定を表示します:

```
> sudo virsh dominfo sles15 | grep memory
Max memory:     8388608 KiB
Used memory:    4194608 KiB
```

2. 8GB までメモリ量を増やします:

```
> sudo virsh setmem sles15 8388608
```

3. 変更が反映されたことを確認します:

```
> sudo virsh dominfo sles15 | grep memory
Max memory:     8388608 KiB
Used memory:    8388608 KiB
```

重要: 巨大なメモリを使用する VM ゲスト の場合について

4TB 以上のメモリを必要とする VM ゲスト を動作させる場合、現時点では `host-passthrough` CPU モードを使用するか、`host-model` または `custom` CPU モードを使用する場合は仮想 CPU アドレスサイズを明示的に指定しなければなりません。これは、後者 2 つの CPU モードの場合、既定の仮想 CPU アドレスサイズでは 4TB 以上のメモリに対して不適切な設定になってしまっているためです。下記の例では、`host-model` CPU モードを使用する際の VM ホストサーバ の物理 CPU アドレスサイズの設定方法を示しています。

```
[...]
<cpu mode='host-model' check='partial'>
<maxphysaddr mode='passthrough'>
</cpu>
[...]
```

仮想 CPU アドレスサイズに関する詳細は、https://libvirt.org/formatdomain.html#cpu-model-and-topology (英語) にある CPU model and topology (CPU モデルとトポロジ) 内の `maxphysaddr` オプションの説明をお読みください。

14.7 大規模 NUMA システムにおける CPU とメモリの設定と配置

複数の NUMA ノードを持つ大規模 VM ホストサーバ において大規模な VM ゲスト を配置する場合、VM ゲスト への割り当てを適切に行うため、性能とリソースの設定を変更する必要があるかもしれません。

14.7.1 huge page

VM ホストサーバ から VM ゲスト に対してメモリを割り当てる際、huge page を使用することで、多くの処理に対する性能を改善できる可能性があります。下記の手順では、VM ゲスト のメモリに対して huge page を設定するための手順を説明しています。なお、VM ゲスト はあらかじめシャットダウンしておくか、設定の変更後に起動し直す必要があることに注意してください。

1. VM ゲスト の設定を編集します:

```
> sudo virsh edit sles15
```

2. `domain` 要素の直下に、下記のような `memoryBacking` 要素を追加します:

```
<memoryBacking>
            <hugePages/>
            <memoryBacking/>
```

3. VM ゲストを起動していない場合は起動します。起動中の場合はいったんシャットダウンして、起動し直してください。

`memoryBacking` 設定について、詳しくは https://libvirt.org/formatdomain.html#memory-backing にある Memory Backing の項をお読みください。

14.7.2 VM ゲストに対する自動 NUMA 配置

大規模な物理 NUMA システムに対して大規模な VM を配置する場合、CPU やメモリリソースが VM ホストサーバの NUMA ノード内を不用意に移動して性能劣化を引き起こしたりしないようにする、という性能改善手法があります。これにより全てのリソースが NUMA ノード内の近い場所に配置できるようになります。また、性能を適切に確保するため、VM ゲストの CPU やメモリの割り当ては、単一の VM ホストサーバ NUMA ノード内に収まるようにしておくことも必要です。

`libvirt` では外部の `numa-preplace` プログラムを利用することで、VM ゲストの CPU とメモリのリソースを自動的に配置することができます。このような自動配置を有効化するには、`cpu` 要素内の `placement` 属性を `auto` に設定します。なお、メモリの配置についても `numatune` 要素で `auto` を指定する必要があります。

1. VM ゲストの設定を編集します:

```
> sudo virsh edit sles15
```

2. トップレベル要素である `domain` 内に `numatune` 要素を追加し、その中に `mode` 属性に `auto` を設定した `memory` 要素を追加します:

```
<numatune>
            <memory mode="auto"/>
            <numatune/>
```

3. VM ゲストを起動していない場合は起動します。起動中の場合はいったんシャットダウンして、起動し直してください。

重要: 自動 NUMA 配置について

自動 NUMA 配置は `numa-preplace` パッケージがインストールされている場合にのみ利用可能となります。また、`numa-preplace` は通常のメモリと huge page の両方で動作します。

`numatune` 設定について、詳しくは https://libvirt.org/formatdomain.html#numa-node-tuning にある NUMA Node Tuning の項をお読みください。

14.8 PCI デバイスの追加

`virsh` を利用して VM ゲストに対して PCI デバイスを追加するには、下記の手順を実施します:

1. まずは VM ゲストに割り当てるホスト側の PCI デバイスを識別します。下記の例では、DEC 社のネットワークカードをゲストに割り当てようとしています:

```
> sudo lspci -nn
[...]
17:00.0 Ethernet controller [0200]: Digital Equipment Corporation DECchip \
21140 [FasterNet] [1011:0009] (rev 22)
[...]
```

   デバイス ID (上記の例では `17:00.0`) をメモしておきます。

2. `virsh nodedev-dumpxml ID` を実行して、デバイスに関する詳細情報を取得します。ここで `ID` にはデバイス ID (この例では `17:00.0`) を指定しますが、コロン (:) とピリオド (.) をアンダースコア (_) に置き換え、かつ「pci_0000_」という前置きを置いた値 (この例では `pci_0000_17_00_0` になります) を指定して実行します:

```
> sudo virsh nodedev-dumpxml pci_0000_17_00_0
<device>
  <name>pci_0000_17_00_0</name>
  <path>/sys/devices/pci0000:00/0000:00:14.4/0000:17:00.0</path>
  <parent>pci_0000_00_14_4</parent>
  <driver>
    <name>tulip</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>23</bus>
    <slot>0</slot>
    <function>0</function>
    <product id='0x0009'>DECchip 21140 [FasterNet]</product>
    <vendor id='0x1011'>Digital Equipment Corporation</vendor>
```

```
    <numa node='0'/>
  </capability>
</device>
```

出力された値の中から、domain, bus, slot, function の値 (上記太字部分) をメモしておきます。

3. VM ゲスト に対して割り当てを行う前に、VM ホストサーバ 側からの切り離しを行います:

```
> sudo virsh nodedev-detach pci_0000_17_00_0
  Device pci_0000_17_00_0 detached
```



ヒント: 多機能型 PCI デバイスについて

FLR (Function Level Reset; 機能レベルリセット) や PM (Power Management; 電源管理) リセットに対応していない多機能型の PCI デバイスを使用している場合、VM ゲスト 側に割り当てるには、VM ホストサーバ 側で全ての機能を切り離す必要があります。また、セキュリティ上の理由から、デバイス全体をリセットする必要があります。`libvirt` では、VM ホストサーバ 側もしくは他の VM ゲスト 側で機能の一部が使用されている場合、その割り当てを拒否するようになっています。

4. domain, bus, slot, function の各値を 16 進数に変換します。上記の例では、domain = 0, bus = 23, slot = 0, function = 0 ですので、下記のようなコマンドを実行して変換を行います。なお、指定の順序を間違えないようにしてください:



注記: `nodedev-dumpxml` の出力が 10 進数である件について

`virsh nodedev-dumpxml` の `<capability>` セクション内にある `domain` , `bus` , `slot` , `function` の各値は、デバイス名 (今回の例では `pci_0000_17_00_0`) や `lspci` での 16 進数出力とは異なり、10 進数であることに注意してください。ここでは `nodedev-dumpxml` の 10 進数部分のみを変換する必要があります。デバイス名や `lspci` の出力を再度 `printf` で 16 進数に変換しなおしてしまうと、誤ったアドレスになってしまいます。たとえばバス番号 `23` の 16 進数表記である `0x17` を再度変換してしまうと、`0x11` という値になってしまいます。

```
> printf "<address domain='0x%x' bus='0x%x' slot='0x%x' function='0x%x'/>
" 0 23 0 0
```

上記を実行すると、下記のように出力されるはずです:

```
<address domain='0x0' bus='0x17' slot='0x0' function='0x0'/>
```

5. あとは対象の VM ゲスト の設定ファイルを `virsh edit` で編集して、上記の手順の出力結果を `<devices>` タグ内に貼り付けます:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0' bus='0x17' slot='0x00' function='0x0'/>
  </source>
</hostdev>
```



ヒント: managed と unmanaged の違いについて

`libvirt` では、PCI デバイスの処理方法を 2 種類 (`managed` と `unmanaged`) 用意しています。`managed` の場合、`libvirt` は必要に応じて既存のドライバからデバイスを切り離す処理からデバイスのリセット、仮想マシンを起動する前の `vfio-pci` への接続など、全ての詳細処理を制御するようになります。また、仮想マシンが終了した場合や仮想マシンからデバイスを切り離した場合、`libvirt` は `vfio-pci` への接続を解除して元のドライバに再接続する処理までを行うようになります。逆に `unmanaged` の場合、`libvirt` はそれらの処理を行わず、仮想マシンにハードウエアを割り当てる際と、仮想マシンからハードウエアを切り離す際には、それらの処理をユーザ側で行わなければなりません。

上記の例では `managed='yes'` を指定しているため、`managed` を選択していることになります。`unmanaged` に切り替えたい場合は、これを `managed='no'` に変更してください。なお、この場合は `virsh nodedev-detach` や `virsh nodedev-reattach` のコマンドを利用して、対応するドライバに対する処理を行う必要があります。具体的には、VM ゲスト を起動する前に `virsh nodedev-detach pci_0000_17_00_0` を実行してホスト側から切り離し、終了後には `virsh nodedev-reattach pci_0000_17_00_0` を実行して、ホスト側で認識できるように再設定する必要があります。

6. ホスト側で SELinux が動作している場合は、VM ゲスト をシャットダウンして SELinux を無効化します。

```
> sudo setsebool -P virt_use_sysfs 1
```

7. VM ゲストを起動すると、VM ゲストから割り当てられた PCI デバイスにアクセスできるようになります:

```
> sudo virsh start sles15
```

重要: SLES11 SP4 の KVM ゲストについて

新しい QEMU マシンタイプ (pc-i440fx-2.0 もしくはそれ以降) を設定した SLES 11 SP4 KVM ゲストの場合、ゲスト内では既定で `acpiphp` モジュールが読み込まれません。このモジュールはディスクやネットワークデバイスのホットプラグ (活性接続) を行うために読み込んでおかなければならないモジュールですので、必要であれば `modprobe acpiphp` コマンドを実行して読み込んでください。なお、`/etc/modprobe.conf.local` ファイル内に `install acpiphp /bin/true` の行を追加すると、システムの起動時に自動読み込みを行うことができます。

重要: QEMU Q35 マシンタイプを使用する KVM ゲストについて

QEMU Q35 マシンタイプを使用する KVM マシンの場合、1 つの `pcie-root` コントローラと 7 つの `pcie-root-port` コントローラからなる PCI トポロジを構成します。`pcie-root` コントローラはホットプラグ (活性接続) には対応しませんが、`pcie-root-port` コントローラはそれぞれ 1 つの PCIe デバイスのホットプラグに対応します。PCI コントローラ自身はホットプラグに対応していませんので、7 つ以上の PCIe デバイスのホットプラグが必要となる場合、あらかじめ `pcie-root-port` コントローラを追加しておくようにしてください。また `pcie-to-pci-bridge` コントローラを追加することで、古い PCI デバイスのホットプラグを実現することもできます。QEMU のマシンタイプ別の PCI トポロジの詳細について、詳しくは https://libvirt.org/pci-hotplug.html (英語) をお読みください。

14.8.1 IBM Z 向けの PCI パススルー

IBM Z をサポートするため、QEMU は追加の属性を設定するために PCI 表記を拡張しています。具体的には `uid` と `fid` という 2 種類の属性が `libvirt` 仕様内の `<zpci/>` に追加されています。`uid` はユーザ定義の識別値を、`fid` の PCI 機能の識別値をそれぞれ設定します。これらの属性はいずれも任意指定で、何も指定しない場合は矛盾の無い値を自動的に生成します。

ドメイン設定内に zPCI 属性を含めたい場合は、下記の例を参考にしてください:

```
<controller type='pci' index='0' model='pci-root'/>
<controller type='pci' index='1' model='pci-bridge'>
  <model name='pci-bridge'/>
```

```
  <target chassisNr='1'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'>
    <zpci uid='0x0001' fid='0x00000000'/>
  </address>
</controller>
<interface type='bridge'>
  <source bridge='virbr0'/>
  <model type='virtio'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x01' function='0x0'>
    <zpci uid='0x0007' fid='0x00000003'/>
  </address>
</interface>
```

14.9 USB デバイスの追加

USB デバイスを VM ゲスト に割り当てるには、`virsh` を使用して下記のように実施します:

1. VM ゲスト に接続されている USB デバイスを識別します:

   ```
   > sudo lsusb
   [...]
   Bus 001 Device 003: ID 0557:2221 ATEN International Co., Ltd Winbond Hermon
   [...]
   ```

   出力された ID をメモしておきます。上記の例では、製造元 ID が `0557` 、製品 ID が `2221` となります。

2. 仮想マシンに対して `virsh edit` を実行し、下記の内容を `<devices>` タグ内に追加します。このとき、vendor と product の箇所にそれぞれメモした値を指定します:

   ```
   <hostdev mode='subsystem' type='usb'>
     <source startupPolicy='optional'>
      <vendor id='0557'/>
      <product id='2221'/>
     </source>
   </hostdev>
   ```



ヒント: 製造元 (vendor) と製品 (product) の指定ではなく、デバイスのアドレス (address) を設定する方法について

`<vendor/>` および `<product/>` の ID を指定する方法のほかにも、14.8項「PCI デバイスの追加」で PCI デバイスを割り当てる際の説明と同様に、`<address/>` タグを利用して割り当てる方法もあります。

3. ホスト内で SELinux が動作している場合は、VM ゲスト をシャットダウンして SELinux を無効化します。

```
> sudo setsebool -P virt_use_sysfs 1
```

4. VM ゲスト を起動すると、VM ゲスト から割り当てられた PCI デバイスにアクセスできるようになります:

```
> sudo virsh start sles15
```

14.10 SR-IOV デバイスの追加

Single Root I/O Virtualization (SR-IOV) に対応した PCIe デバイスは、リソースを複製することができるため、複数のデバイスとして振る舞うことができます。複製されたリソースは「擬似デバイス」として、VM ゲスト への割り当てを行うことができます。

SR-IOV は Peripheral Component Interconnect Special Interest Group (PCI-SIG) が作成した工業仕様で、物理機能 (Physical Functions (PF)) と仮想機能 (Virtual Functions (VF)) を提供しています。PF はデバイスを管理したり設定したりするための完全な PCIe 機能で、データの移動も行うことができます。それに対して VF 側には管理部分が提供されておらず、データの移動と設定機能の一部のみが提供されています。VF は全ての PCIe 機能を持っているわけではないので、ホスト側のオペレーティングシステムもしくは ハイパーバイザ が SR-IOV に対応し、VF へのアクセスと初期化を行わなければなりません。論理上の VF の最大数は、1 デバイスあたり 256 個まで (たとえば 2 ポートのイーサネットカードであれば 512 個) になります。実際には各 VF がリソースを消費してしまうことから、この最大値はもっとずっと小さくなります。

14.10.1 要件

SR-IOV を使用するには、下記の要件を全て満たさなければなりません:

- SR-IOV に対応したネットワークカードを用意すること (openSUSE Leap ではネットワークカードのみに対応しています) 。
- AMD64/Intel 64 でハードウエア仮想化 (AMD-V もしくは Intel VT-x) に対応していること。
- デバイスの割り当て (AMD-Vi もしくは Intel VT-d) に対応したチップセットであること。
- `libvirt` 0.9.10 もしくはそれ以降が存在すること。
- ホストシステム内で SR-IOV ドライバが読み込まれ、設定されていること。

- ホスト側の設定が 重要: VFIO および SR-IOV の要件について に示されている要件を満たしていること。
- VM ゲスト に割り当てる予定の VF の PCI アドレスの一覧を用意していること。



ヒント: デバイスが SR-IOV に対応しているかどうかの確認方法

デバイスが SR-IOV に対応しているかどうかは、`lspci -v` を実行して表示される情報から判断することができます。SR-IOV に対応するデバイスである場合、下記のような表示が現れるはずです:

```
Capabilities: [160 v1] Single Root I/O Virtualization (SR-IOV)
```



注記: VM ゲスト の作成時における SR-IOV デバイスの追加について

初期設定時に VM ゲスト に SR-IOV デバイスを追加する場合は、あらかじめ 14.10.2項 「SR-IOV ホストドライバの読み込みと設定」で説明している手順で設定を済ませておく必要があります。

14.10.2 SR-IOV ホストドライバの読み込みと設定

VF にアクセスして準備を行うには、SR-IOV 対応のドライバをホスト側のシステムに読み込んでおく必要があります。

1. ドライバを読み込む前に、まずは `lspci` を実行して、カードが正しく検出されていることを確認します。下記の例では、`lspci` がデュアルポートの Intel 82576NS ネットワークカードを検出しています:

```
> sudo /sbin/lspci | grep 82576
01:00.0 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
01:00.1 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
04:00.0 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
04:00.1 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
```

カードが検出されていない場合、BIOS/EFI の設定でハードウエア仮想化の設定が有効化されていないことが考えられます。ハードウエア仮想化機能が有効化されているか同化を調べるには、ホスト側の BIOS 設定をご確認ください。

2. 次に `lsmod` を実行して、SR-IOV ドライバが読み込まれているかどうかを確認します。下記の例では igb ドライバ (Intel 82576NS ネットワークカード向けのドライバです) が読み込まれているかどうかの確認になります。下記のように表示されれば、ドライバが既に読み込まれていることになります。何も出力を返さない場合は、ドライバが読み込まれていないことになります。

```
> sudo /sbin/lsmod | egrep "^igb "
igb                   185649  0
```

3. ドライバが既に読み込まれている場合は、この手順を飛ばしてください。SR-IOV ドライバが読み込まれていない場合は、あらかじめ SR-IOV 非対応のドライバの読み込みを解除する必要があります。読み込みを解除するには、`rmmod` コマンドをお使いください。下記の例では、Intel 82576NS ネットワークカード向けの SR-IOV 非対応ドライバの読み込みを解除しています:

```
> sudo /sbin/rmmod igbvf
```

4. あとは `modprobe` を利用して、SR-IOV 対応のドライバを読み込みます。このとき、VF パラメータ (`max_vfs`) を必ず指定してください:

```
> sudo /sbin/modprobe igb max_vfs=8
```

代わりの方法として、SYSFS を介してドライバを読み込む方法もあります:

1. イーサネットデバイスの一覧を表示して、物理 NIC の PCI ID を確認します:

```
> sudo lspci | grep Eth
06:00.0 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev 10)
06:00.1 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev 10)
```

2. VF を有効化するには、`sriov_numvfs` パラメータに対して必要な VF 数を書き込みます:

```
> sudo echo 1 > /sys/bus/pci/devices/0000:06:00.1/sriov_numvfs
```

3. VF NIC が読み込まれたことを確認します:

```
> sudo lspci | grep Eth
06:00.0 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev 10)
06:00.1 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev 10)
06:08.0 Ethernet controller: Emulex Corporation OneConnect NIC (Skyhawk) (rev 10)
```

4. 設定可能な VF の最大数を知りたい場合は、下記のようなコマンドを入力して実行します:

```
> sudo lspci -vvv -s 06:00.1 | grep 'Initial VFs'
                      Initial VFs: 32, Total VFs: 32, Number of VFs: 0,
Function Dependency Link: 01
```

5. `/etc/systemd/system/before.service` ファイルを作成して、システムの起動時に SYSFS 経由で VF を自動設定するように設定します:

```
[Unit]
Before=
[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/bash -c "echo 1 > /sys/bus/pci/devices/0000:06:00.1/sriov_numvfs"
# 注意: 実行ファイルはシェル経由ではなく、直接実行されます。詳しい書式については、
# systemd.service と systemd.unit のマニュアルページをお読みください
[Install]
# このサービスを開始するターゲットの指定
WantedBy=multi-user.target
#WantedBy=graphical.target
```

6. また VM を起動する前に、もう 1 つのサービスファイル（ `after-local.service` ）を作成し、NIC の切り離しを行うスクリプト `/etc/init.d/after.local` を実行するように設定します。これを行わないと、VM の起動が失敗するようになってしまいます:

```
[Unit]
Description=/etc/init.d/after.local Compatibility
After=libvirtd.service
Requires=libvirtd.service
[Service]
Type=oneshot
ExecStart=/etc/init.d/after.local
RemainAfterExit=true

[Install]
WantedBy=multi-user.target
```

7. 上記の内容を `/etc/systemd/system` にコピーします。

```
#! /bin/sh
# ...
virsh nodedev-detach pci_0000_06_08_0
```

上記の内容を `/etc/init.d/after.local` に保存します。

8. マシンを再起動したあと、手順の冒頭で実行した `lspci` コマンドを実行しなおし、SR-IOV ドライバが読み込まれていることを確認します。SR-IOV ドライバが正しく読み込まれていれば、下記のように VF 向けの追加の行が現れているはずです:

```
01:00.0 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
01:00.1 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
01:10.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
01:10.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
01:10.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
[...]
04:00.0 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
04:00.1 Ethernet controller: Intel Corporation 82576NS Gigabit Network Connection
 (rev 01)
04:10.0 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
04:10.1 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
04:10.2 Ethernet controller: Intel Corporation 82576 Virtual Function (rev 01)
[...]
```

14.10.3 VM ゲスト に対する VF ネットワークデバイスの追加

SR-IOV 対応のハードウエアの設定を VM ホストサーバ 内で正しく実施したあとは、VM ゲスト に対して VF を追加していきます。これを行うには、まずいくつかのデータを収集しておく必要があります。

手順 14.2: 既存の VM ゲスト に対する VF ネットワークデバイスの追加

下記に示す手順は構成例となります。お使いの環境に合わせて各種のデータを変更して実行してください。

1. `virsh nodedev-list` コマンドを実行して、割り当てる VF と対応する PF の PCI アドレスを取得します。`lspci` の出力は 14.10.2項「SR-IOV ホストドライバの読み込みと設定」のようになります (たとえば `01:00.0` や `04:00.1` など) 。このアドレス情報のコロン (:) やドット (.) をアンダースコア (_) に変換し、冒頭に "pci_0000_" を付けたものが virsh で使用するアドレスとなります。たとえば `lspci` コマンドで "04:00.0" と出力された場合、virsh のアドレスは "pci_0000_04_00_0" になります。下記の例では、Intel 82576NS デュアルポートイーサネットカードでの PCI ID を取得しています:

```
> sudo virsh nodedev-list | grep 0000_04_
pci_0000_04_00_0
pci_0000_04_00_1
pci_0000_04_10_0
pci_0000_04_10_1
pci_0000_04_10_2
```

```
pci_0000_04_10_3
pci_0000_04_10_4
pci_0000_04_10_5
pci_0000_04_10_6
pci_0000_04_10_7
pci_0000_04_11_0
pci_0000_04_11_1
pci_0000_04_11_2
pci_0000_04_11_3
pci_0000_04_11_4
pci_0000_04_11_5
```

最初の 2 つの項目が `PF` を、残りの項目が VF を表しています。

2. あとは `virsh nodedev-dumpxml` を実行して、追加したい VF の PCI ID を取得します:

```
> sudo virsh nodedev-dumpxml pci_0000_04_10_0
<device>
  <name>pci_0000_04_10_0</name>
  <parent>pci_0000_00_02_0</parent>
  <capability type='pci'>
    <domain>0</domain>
    <bus>4</bus>
    <slot>16</slot>
    <function>0</function>
    <product id='0x10ca'>82576 Virtual Function</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <capability type='phys_function'>
      <address domain='0x0000' bus='0x04' slot='0x00' function='0x0'/>
    </capability>
  </capability>
</device>
```

次の手順では、下記のデータが必要となります:

- `<domain>0</domain>`
- `<bus>4</bus>`
- `<slot>16</slot>`
- `<function>0</function>`

3. あとは一時的な XML ファイル (たとえば `/tmp/vf-interface.xml`) を作成して、既存の VM ゲスト に VF ネットワークデバイスを追加するのに必要なデータを記述していきます。このファイルを最も短く作成すると、下記のようになります:

```
<interface type='hostdev'>❶
 <source>
```

```
  <address type='pci' domain='0' bus='11' slot='16' function='0'2/>❷
 </source>
</interface>
```

❶ VF は固定の MAC アドレスを取得しません。ホストが再起動されるたびに MAC アドレスが変更される形になります。この場合、`hostdev` で「従来の方法を利用して」ネットワークデバイスを追加すると、ホスト側の再起動が発生するたびに MAC アドレスが変わってしまうため、VM ゲスト 側のネットワークデバイスを再設定する必要が生じてしまいます。このような問題を回避するため、`libvirt` では `hostdev` という値が提供されるようになり、これによってデバイスの割り当て 前 にネットワーク固有のデータを設定できるようになっています。

❷ 以前の手順で取得したデータを指定します。

4. デバイスが既にホスト側に割り当てられてしまっている場合、VM ゲスト に対して割り当てることができなくなります。ゲストに対して割り当てたい場合は、下記のようにしてホスト側から切り離しを行ってください:

```
> sudo virsh nodedev-detach pci_0000_04_10_0
```

5. あとは既存の VM ゲスト に VF インターフェイスを追加します:

```
> sudo virsh attach-device 仮想マシン名 /tmp/vf-interface.xml --オプション
```

仮想マシン名 の箇所には VM ゲスト の名前のほか、ID や UUID を指定することもできます。また、-- オプション の部分には下記を指定することができます:

`--persistent`
: 仮想マシンの XML 設定ファイルに対してデバイスを追加します。これに加えて、仮想マシンが動作中である場合、ホットプラグで接続することもできます。

`--config`
: 仮想マシンの XML 設定ファイルに対してデバイスを追加します。仮想マシンが動作中の場合、ホットプラグは行われず、次回の再起動以降に現れるようになります。

`--live`
: 仮想マシンの動作中の状態にのみ適用します。仮想マシンが動作中ではない場合、この操作は失敗します。XML ファイル内には保存されませんので、VM ゲスト の再起動を行うとデバイスが消えてしまいます。

`--current`

仮想マシンの現在の状態に反映させます。仮想マシンが動作中ではない場合、デバイスを XML 設定ファイル内に追加し、次回の起動時に現れるようになります。仮想マシンが動作中である場合、デバイスはホットプラグで追加されますが、XML 設定ファイルには追加されないようになります。

6. VF インターフェイスを切り離すには、`virsh detach-device` コマンドを使用します。オプション類は上記と同じです。

14.10.4 プールからの動的な VF の割り当て

14.10.3項「VM ゲスト に対する VF ネットワークデバイスの追加」の手順に従って VM ゲスト の設定内に VF の PCI アドレスを指定してしまうと、他のホストへの移行が難しくなってしまいます。移行先のホストで移行元と同じ PCI バスに同じハードウエアが搭載されていれば問題はありませんが、そうでない場合は VM ゲスト の設定を変更しなければならなくなってしまいます。

このような場合は、SR-IOV の全ての VF を含むデバイスプールを設定し、`libvirt` 側から使用できるようにする方法があります。VM ゲスト では起動時にこのプールを参照し、空いているいずれかのデバイスを動的に使用することができます。VM ゲスト を停止すると VF はプール内に戻されますので、他のゲストから使用できるようになります。

14.10.4.1 VM ホストサーバ 内の VF プールを利用したネットワークの定義

下記の例では、ホスト側で `eth0` のネットワークインターフェイスが割り当てられている PF に対して、それに結びつく全ての SR-IOV デバイスである VF のプールを作成しています:

```
<network>
  <name>passthrough</name>
    <forward mode='hostdev' managed='yes'>
      <pf dev='eth0'/>
    </forward>
  </network>
```

このネットワークインターフェイスをホスト側で使用する場合は、上記のコードをファイルに保存 (例: `/tmp/passthrough.xml`) したあと、下記のコマンドを実行してください。なお、上記の `eth0` の箇所はお使いの環境内の SR-IOV デバイスの PF に置き換えてください:

```
> sudo virsh net-define /tmp/passthrough.xml
> sudo virsh net-autostart passthrough
> sudo virsh net-start passthrough
```

14.10.4.2 プールから VF を使用するための VM ゲスト 側の設定

下記の VM ゲスト のデバイスインターフェイス定義は、14.10.4.1項「VM ホストサーバ 内の VF プールを利用したネットワークの定義」で作成したプールから SR-IOV デバイスの VF を使用する設定です。`libvirt` では、最初のゲストの起動時に、PF に結びつけられた VF の一覧を自動的に取得します。

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

VF のプールからネットワークインターフェイスを使用するように設定した VM ゲスト を起動したあとは、VF が正しく使用されていることを確認します。これを行うには、ホスト側で `virsh net-dumpxml passthrough` を実行します:

```
<network connections='1'>
  <name>passthrough</name>
  <uuid>a6a26429-d483-d4ed-3465-4436ac786437</uuid>
  <forward mode='hostdev' managed='yes'>
    <pf dev='eth0'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x1'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x3'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x5'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x7'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x1'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x3'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x5'/>
  </forward>
  </network>
```

14.11 接続されているデバイスの一覧表示

`virsh` には VM ゲスト に接続されている全ての VM ホストサーバ のデバイスを一覧表示する機能はありませんが、指定した VM ゲスト に接続されているデバイスを一覧表示することはできます。具体的には下記のようなコマンドを入力して実行します:

```
virsh dumpxml VM_ゲスト名 | xpath -e /domain/devices/hostdev
```

たとえば下記のようになります:

```
> sudo virsh dumpxml sles12 | -e xpath /domain/devices/hostdev
Found 2 nodes:
-- NODE --
<hostdev mode="subsystem" type="pci" managed="yes">
```

```
  <driver name="xen" />
  <source>
    <address domain="0x0000" bus="0x0a" slot="0x10" function="0x1" />
  </source>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x0a" function="0x0" />
  </hostdev>
-- NODE --
<hostdev mode="subsystem" type="pci" managed="yes">
  <driver name="xen" />
  <source>
    <address domain="0x0000" bus="0x0a" slot="0x10" function="0x2" />
  </source>
  <address type="pci" domain="0x0000" bus="0x00" slot="0x0b" function="0x0" />
</hostdev>
```



ヒント: <interface type='hostdev'> で接続されている SR-IOV デバイスの一覧表示について

`<interface type='hostdev'>` を利用して VM ホストサーバ のデバイスに接続している SR-IOV デバイスの場合は、上記とは異なる XPath クエリを指定する必要があります:

```
virsh dumpxml VM_ゲスト名 | xpath -e /domain/devices/interface/@type
```

14.12 ストレージデバイスの設定

ストレージデバイスは `disk` 内に記述します。通常、`disk` タグには複数の属性を指定します。そのうち、下記に示す 2 つの属性が最も重要です:

- `type` 属性: 仮想ディスクデバイスのソースを指定するための属性です。`file` , `block` , `dir` , `network` , `volume` のいずれかを指定します。
- `device` 属性: ディスクを VM ゲスト 側の OS に示す際の方法を指定する属性です。`floppy` (フロッピィディスク), `disk` (ハードディスク), `cdrom` (CD-ROM) などがあります。

子要素として最も重要なものは下記のとおりです:

- `driver` にはドライバとバスに関する情報を指定します。ここには VM ゲスト 側での処理方法などを指定します。
- `target` には VM ゲスト 内でのデバイス名の設定が含まれます。ここには任意指定の bus 属性 (接続されているストレージバスに関する情報) を含むことができます。

VM ゲスト にストレージデバイスを追加するには、下記の手順を実施します:

1. 既存の VM ゲスト に対する設定を編集します:

```
> sudo virsh edit sles15
```

2. `devices` 要素内に `disk` 要素を追加し、その中に `type` と `device` の属性を追加します。

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

3. `driver` タグでは下記のような既定値を指定します:

```
<driver name='qemu' type='qcow2'/>
```

4. 新しい仮想ディスクデバイスのソースとなるディスクイメージを作成します:

```
> sudo qemu-img create -f qcow2 /var/lib/libvirt/images/sles15.qcow2 32G
```

5. あとは `source` タグでソースを指定します:

```
<source file='/var/lib/libvirt/images/sles15.qcow2'/>
```

6. VM ゲスト 内でのデバイス名を表す `target` タグを追加します。このとき、接続先を表す `bus` 属性も設定しておきます:

```
<target dev='vda' bus='virtio'/>
```

7. あとは仮想マシンを再起動します:

```
> sudo virsh start sles15
```

これで、VM ゲスト の OS 内から新しいストレージデバイスにアクセスできるようになります。

14.13 コントローラデバイスの設定

`libvirt` では一般に、VM ゲスト が使用する仮想デバイスの種類に応じて、自動的にコントローラを管理することができます。たとえば VM ゲスト 内に PCI デバイスや SCSI デバイスが存在する場合、PCI や SCSI のコントローラは自動的に作成され管理されます。このほか、`libvirt` ではハイパーバイザ固有のコントローラ、たとえば KVM の VM ゲスト であれば `virtio-serial` 、Xen の VM ゲスト であれば `xenbus` を作成することができます。通常は既定のコントローラとそれに付随する設定で問題なく動作しますが、必要に応じてコントローラやその設定を調整することもできます。たとえば virtio-serial でより多くのポートが必要となっている場合や、xenbus コントローラに対してより多くのメモリを割り当てたり、割り込みを多く割り当てたりするような場合がそれに該当します。

xenbus コントローラは、全ての Xen 準仮想化デバイスに対するコントローラとして動作する点でユニークなものであると言えます。また、VM ゲスト に数多くのディスクやネットワークデバイスが接続されるような場合には、コントローラに対してもメモリを多く割り当てる必要があるかもしれません。Xen の `max_grant_frames` 属性は許可するフレームの数、もしくは共有メモリのブロック数を指定しますが、これはそれぞれの VM ゲスト に対する `xenbus` コントローラに対して割り当てられます。

既定値は 32 で、ほとんどの環境において十分な値ではありますが、I/O デバイスの多い VM ゲストや I/O 負荷の高いシステムでは、フレームの枯渇によって性能が落ちることがあります。このような場合、`xen-diag` コマンドを利用して dom0 およびお使いの VM ゲスト の `max_grant_frames` の現在値および最大値を確認してください。なお、ゲストは動作中でなければなりません:

```
> sudo virsh list
 Id   Name             State
--------------------------------
 0    Domain-0         running
 3    sle15sp1         running

 > sudo xen-diag gnttab_query_size 0
domid=0: nr_frames=1, max_nr_frames=256

> sudo xen-diag gnttab_query_size 3
domid=3: nr_frames=3, max_nr_frames=32
```

上記では、`sle15sp1` のゲストは 32 フレームのうちの 3 フレームしか使用していません。性能面で何らかの問題がある場合や、フレーム数が不足している旨のログが記録されているような場合は、`virsh` でフレーム数を増やしてください。具体的には、ゲスト側の設定ファイル内にある `<controller type='xenbus'` という行を探して、`maxGrantFrames` という制御要素を追加します:

```
> sudo virsh edit sle15sp1
 <controller type='xenbus' index='0' maxGrantFrames='40'/>
```

設定を保存してゲストを再起動してください。これで設定が反映されます:

```
> sudo xen-diag gnttab_query_size 3
domid=3: nr_frames=3, max_nr_frames=40
```

maxGrantFrames と同様に、xenbus コントローラには `maxEventChannels` という設定値も用意されています。それぞれのチャンネルは準仮想化された割り込みのような動作をするもので、許可するフレーム数と同時に、準仮想化ドライバ向けのデータ転送構造を構築するためのものです。仮想 CPU 数の多い VM ゲスト や多数の準仮想化デバイスが接続されている VM ゲスト の場合は、既定値である 1023 よりも大きい値を設定する必要があるかもしれません。maxEventChannels は maxGrantFrames と同じように変更することができます:

```
> sudo virsh edit sle15sp1
 <controller type='xenbus' index='0' maxGrantFrames='128' maxEventChannels='2047'/>
```

詳しくは https://libvirt.org/formatdomain.html#elementsControllers (英語) にある libvirt Domain XML format マニュアルの中にある、Controllers 章をお読みください。

14.14 ビデオデバイスの設定

仮想マシンマネージャを利用した場合、ビデオデバイスのモデルのみを設定することができます。VRAM の割り当てや 2D/3D のアクセラレーションの設定については、XML の設定を編集することでしか設定することができません。

14.14.1 VRAM の割当量の変更

1. 既存の VM ゲスト に対する設定を編集します:

```
> sudo virsh edit sles15
```

2. VRAM に割り当てるサイズを変更するには、下記のように設定します:

```
<video>
<model type='vga' vram='65535' heads='1'>
...
</model>
</video>
```

3. あとは仮想マシンマネージャを開いて、仮想マシン側に割り当てられた VRAM のサイズを確認してください。

14.14.2 2D/3D アクセラレーションの設定変更

1. 既存の VM ゲスト に対する設定を編集します:

```
> sudo virsh edit sles15
```

2. 2D/3D アクセラレーション機能を有効化／無効化するには、それぞれ `accel3d` もしくは `accel2d` の設定を変更します:

```
<video>
 <model>
  <acceleration accel3d='yes' accel2d='no'>
 </model>
</video>
```



ヒント: 2D/3D アクセラレーションの有効化について

2D/3D アクセラレーション機能を使用するには、`virtio` もしくは `vbox` ビデオデバイスを使用する必要があります。それ以外のビデオデバイスである場合は、機能を有効化することができません。

14.15 ネットワークデバイスの設定

本章では、`virsh` を利用して仮想ネットワークデバイスを設定するための方法について説明しています。

`libvirt` のネットワークインターフェイス仕様について、詳しくは https://libvirt.org/formatdomain.html#elementsDriverBackendOptions をお読みください。

14.15.1 マルチキュー型の virtio-net によるネットワーク性能の強化

マルチキュー型の virtio-net 機能を使用することで、VM ゲスト の仮想 CPU を複数個同時に使用することができるようになります。これにより、ネットワークの性能を改善することができるようになります。一般的な情報については 34.3.3項「マルチキュー型 virtio-net を利用したネットワーク性能の強化」をお読みください。

特定の VM ゲスト に対して virtio-net のマルチキュー設定を行うには、14.1項「VM の設定変更」で示している手順で XML ファイルを編集します。具体的には、下記の箇所を修正します:

```
<interface type='network'>
 [...]
 <model type='virtio'/>
 <driver name='vhost' queues='キュー数'/>
</interface>
```

14.16 VM ホストサーバ のネットワークインターフェイスを共有するための macvtap の使用

macvtap は VM ゲスト の仮想インターフェイスをホストのネットワークインターフェイスに直接結びつけるための方法です。macvtap ベースのインターフェイスは VM ホストサーバ のネットワークインターフェイスを拡張するための仕組みで、同じイーサネットセグメント内で別の MAC アドレスを使用します。通常は VM ゲスト と VM ホストサーバ をそれぞれ同じスイッチに接続する形になります。



注記: Linux ブリッジでは macvtap を使用することができない問題について

macvtap は対象のインターフェイスが Linux ブリッジに接続されている場合、使用することができません。macvtap インターフェイスを作成する前に、ブリッジからインターフェイスを削除しておく必要があります。



注記: macvtap を利用した VM ゲスト と VM ホストサーバ の通信

macvtap を使用すると、VM ゲスト 同士のほか、ネットワーク内の他のホストとも通信を行うことができるようになります。ただし、VM ゲスト が動作している VM ホストサーバ との間は、通信を行うことができません。これは macvtap の意図的な動作によるもので、VM ホストサーバ の物理イーサネットが macvtap ブリッジに割り当てられているためです。VM ゲスト からブリッジへのトラフィックはそのまま物理インターフェイスに送信され、VM ホストサーバ の IP スタックに戻ることができないようになっています。これと同様に、VM ホストサーバ の IP スタックからのトラフィックも物理インターフェイスにそのまま送信され、VM ゲスト の macvtap に戻ることができなくなっています。

macvtap ベースの仮想ネットワークインターフェイスは libvirt でサポートされていて、インターフェイスの種類 (type) を `direct` にすることによって実現することができます。たとえば下記のようになります:

```
<interface type='direct'>
   <mac address='aa:bb:cc:dd:ee:ff'/>
   <source dev='eth0' mode='bridge'/>
   <model type='virtio'/>
   </interface>
```

macvtap の操作モードは、`mode` 属性を設定することで制御することができます。下記の一覧には、設定可能な値とそれらの説明を示しています:

- `vepa` : 全ての VM ゲスト パケットを外部ブリッジに送信します。同じ VM ホストサーバ 内の VM ゲスト を宛先とするパケットは、VEPA 対応ブリッジによって VM ホストサーバ 側に戻される動作になります (現在のブリッジは一般に VEPA 対応ではありません) 。
- `bridge` : 同じ VM ホストサーバ を宛先とするパケットは、ターゲットの macvtap デバイスに直接配信されるようになります。送信元と送信先のデバイスは、直接配送に対応する `bridge` モードである必要があります。いずれかのモードが `vepa` である場合は、VEPA 対応ブリッジが必要になります。

- `private`：全てのパケットを外部ブリッジに送信し、外部のルータやゲートウエイから送信されたパケットは同じ VM ホストサーバ 内のターゲットの VM ゲスト にのみ配信されるようになります。この処理は送信元と送信先のいずれかがプライベートモードである場合の動作になります。
- `passthrough`：ネットワークインターフェイスに対してさらなる力を与えるための特殊なモードです。全てのパケットはインターフェイスに転送され、virtio VM ゲスト では MAC アドレスの変更やプロミスキャスモードに対応することで、インターフェイスのブリッジや VLAN インターフェイスの作成に対応するようになります。ただし、`passthrough` モードではネットワークインターフェイスを共有することはできません。VM ゲスト にインターフェイスを割り当てると、VM ホストサーバ 側からは切り離されます。このような理由から、SR-IOV 仮想機能は VM ゲスト の `passthrough` モードと比較されます。

14.17 メモリバルーンデバイスの無効化

メモリバルーン機能は KVM では既定のオプションになっています。VM ゲスト 側には明示的にデバイスが割り当てられるようになっていますので、VM ゲスト の XML 設定ファイルには、メモリバルーンのタグを追加する必要はありません。何らかの理由で VM ゲスト のメモリバルーン機能を無効化したい場合は、下記のようにして `model='none'` を指定する必要があります:

```
<devices>
   <memballoon model='none'/>
</device>
```

14.18 マルチモニタ (デュアルヘッド) の設定

`libvirt` では、VM ゲスト に対して複数のモニタを接続してそれぞれにビデオ出力を行うことのできる、デュアルヘッド設定に対応しています。

重要: Xen がサポート対象外である点について

Xen ハイパーバイザは、デュアルヘッド設定に対応していません。

手順 14.3: デュアルヘッドの設定

1. 仮想マシンを動作させた状態で、VM ゲスト 内に `xf86-video-qxl` パッケージがインストールされていることを確認します:

   ```
   > rpm -q xf86-video-qxl
   ```

2. VM ゲストをシャットダウンして、14.1項「VM の設定変更」で示している手順で XML ファイルを変更します。

3. このとき、仮想グラフィックカードの型式 (モデル) を 'qxl' にしてください:

```
<video>
 <model type='qxl' ... />
```

4. グラフィックカードの設定内にある `heads` パラメータを増やして、デュアルヘッドの設定を行います。たとえば既定値の `1` から `2` に増やします:

```
<video>
 <model type='qxl' ram='65536' vram='65536' vgamem='16384' heads='2' primary='yes'/>
 <alias name='video0'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'/>
</video>
```

5. VNC ではなく、Spice ディスプレイを使用するように仮想マシンを設定します:

```
<graphics type='spice' port='5916' autoport='yes' listen='0.0.0.0'>
 <listen type='address' address='0.0.0.0'/>
</graphics>
```

6. 仮想マシンを起動して、`virt-viewer` 経由でディスプレイに接続します。たとえば下記のように入力して実行します:

```
> virt-viewer --connect qemu+ssh://ユーザ名@ホスト名/system
```

7. VM の一覧の中から設定を変更した仮想マシンを選択して、[接続] を押します。

8. VM ゲスト内でグラフィカルサブシステム (Xorg) が読み込まれたら、[表示] › [ディスプレイ] › [ディスプレイ 2] を選択すると、2 つめのモニタの出力を表示することができるようになります。

14.19 IBM Z における KVM ゲストへの暗号化アダプタのパススルー

14.19.1 概要

IBM Z には、乱数生成やデジタル署名の検証、暗号化などの便利な機能を提供する暗号化ハードウエアが含まれています。KVM では、これらの暗号化ハードウエアをゲストに占有させることで、パススルーデバイスとして使用することができます。このパススルー機能を設定した場合、ゲストとデバイスとの間の通信をハイパーバイザから監視することはできなくなります。

14.19.2 カバーされる範囲

ここでは、IBM Z ハードウエアで KVM ゲストに対する暗号化アダプタの占有方法を説明しています。手順は下記のような流れになっています:

- まずはホスト側で暗号化アダプタとドメインを既定のドライバからマスクします。
- `vfio-ap` ドライバを読み込みます。
- `vfio-ap` ドライバに対して、暗号化アダプタとドメインを割り当てます。
- あとは暗号化アダプタを使用するようゲスト側を設定します。

14.19.3 要件

- QEMU / `libvirt` の仮想化環境を正しくインストールしておき、問題なく動作できるようにしておく必要があります。
- ホスト側のオペレーティングシステムで、`vfio_ap` と `vfio_mdev` の各モジュールを用意しておきます。

14.19.4 KVM ホスト側での暗号化ドライバの占有設定

1. まずはホスト側で `vfio_ap` と `vfio_mdev` のカーネルモジュールが読み込まれていることを確認します:

```
> lsmod | grep vfio_
```

どちらか 1 つでも読み込まれていない場合は、下記のようにして手作業で読み込みを行います:

```
> sudo modprobe vfio_mdev
```

2. ホスト内で新しい MDEV デバイスを作成し、それが追加されていることを確認します:

```
uuid=$(uuidgen)
$ echo ${uuid} | sudo tee /sys/devices/vfio_ap/matrix/mdev_supported_types/vfio_ap-
passthrough/create
dmesg | tail
[...]
[272197.818811] iommu: Adding device 24f952b3-03d1-4df2-9967-0d5f7d63d5f2 to group 0
[272197.818815] vfio_mdev 24f952b3-03d1-4df2-9967-0d5f7d63d5f2: MDEV: group_id = 0
```

3. 次に KVM ゲストに占有させるホスト側の論理パーティション内のデバイスを識別します:

```
> ls -l /sys/bus/ap/devices/
[...]
lrwxrwxrwx 1 root root 0 Nov 23 03:29 00.0016 -> ../../../devices/ap/card00/00.0016/
lrwxrwxrwx 1 root root 0 Nov 23 03:29 card00 -> ../../../devices/ap/card00/
```

上記の例では、カードとキューがそれぞれ `0` と `16` になっていることがわかります。Hardware Management Console (HMC) の設定とあわせるため、16 進数表記 `16` を 10 進数表記 `22` に変換してください。

4. 対称のアダプタを `zcrypt` から使用されないようにマスクします:

```
> lszcrypt
CARD.DOMAIN TYPE MODE STATUS REQUEST_CNT
-----------------------------------------------
00 CEX5C CCA-Coproc online 5
00.0016 CEX5C CCA-Coproc online 5
```

アダプタをマスクします:

```
> cat /sys/bus/ap/apmask
0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
echo -0x0 | sudo tee /sys/bus/ap/apmask
0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

ドメインをマスクします:

```
> cat /sys/bus/ap/aqmask
0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
echo -0x0 | sudo tee /sys/bus/ap/aqmask
0xfffffdffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

5. アダプタ 0, ドメイン 16 (10 進数で 22) を `vfio-ap` に割り当てます:

```
> sudo echo +0x0 > /sys/devices/vfio_ap/matrix/${uuid}/assign_adapter
> echo +0x16 | sudo tee /sys/devices/vfio_ap/matrix/${uuid}/assign_domain
> echo +0x16 | sudo tee /sys/devices/vfio_ap/matrix/${uuid}/assign_control_domain
```

6. 設定した matrix を確認します:

```
> cat /sys/devices/vfio_ap/matrix/${uuid}/matrix
00.0016
```

7. あとは新しい VM を作成 (詳しくは 第9章「ゲストのインストール」をお読みください) して準備が完了するまで待つか、既存の VM に設定を行います。いずれの場合にしても、VM はシャットダウンしておく必要があります。

8. MDEV デバイスを使用するよう設定を変更します:

```
> sudo virsh edit VM_NAME
[...]
<hostdev mode='subsystem' type='mdev' model='vfio-ap'>
 <source>
  <address uuid='24f952b3-03d1-4df2-9967-0d5f7d63d5f2'/>
 </source>
</hostdev>
[...]
```

9. あとは仮想マシンを再起動します:

```
> sudo virsh reboot VM_名
```

10. ゲストにログインして、アダプタが存在していることを確認します:

```
> lszcrypt
CARD.DOMAIN TYPE MODE STATUS REQUEST_CNT
-----------------------------------------------
00 CEX5C CCA-Coproc online 1
00.0016 CEX5C CCA-Coproc online 1
```

14.19.5 さらに詳しい情報

- 仮想化コンポーネントのインストールについては 第6章「仮想化コンポーネントのインストール」をお読みください。
- `vfio_ap` の構造については https://www.kernel.org/doc/Documentation/s390/vfio-ap.txt (英語) をお読みください。

- 概要および詳細な手順については、https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1787405 (英語) をお読みください。
- VFIO 仲介型デバイス (VFIO Mediated devices (MDEVs)) の構造については https://www.kernel.org/doc/html/latest/driver-api/vfio-mediated-device.html をお読みください。

15 AMD SEV-SNP による仮想マシンのセキュリティ強化

改訂履歴

2026-03-12

AMD SEV-ES に対する説明のうち、"廃止" を "無効化" に修正。

2026-03-03

AMD SEV-ES に対する廃止告知を追加。

2025-06-12

AMD 社が提供する Secure Encrypted Virtualization-Secure Nested Paging (SEV-SNP) を使用することで、お使いの仮想マシンに対するセキュリティを強化することができます。AMD SEV-SNP 機能は仮想マシンをホストシステムや他の仮想マシンから切り離し、データやコードの保護を提供する仕組みです。この機能ではデータを暗号化するほか、仮想マシン内で実行するコードやデータを検出して追跡する仕組みを備えています。この仕組みにより仮想マシンを分離することができますので、たとえ仮想マシンが不正侵入された場合であっても、他の仮想マシンやホストマシンに影響が無いようにすることができます。



注記: AMD SEV-ES の無効化について

AMD SEV-ES は AMD SEV-SNP 技術をサポートする CPU では無効化されています。これは SEV-ES (Encrypted State) が SEV-SNP (Secure Nested Paging) のサブセットであるためで、ハードウエアが SNP に対応していれば、ES の保護に加えてメモリの一貫性保護も行われる新しい標準として動作するためです。

本章では、お使いの AMD EPYC サーバで openSUSE Leap 15.7 を利用した際に、AMD SEV-SNP 機能を有効化するための手順を説明しています。

15.1 対応するハードウエア

AMD SEV-SNP 仮想マシンを実行するには、AMD 社の EPYC (第 3 世代もしくはそれ以降) を搭載したシステムが必要となります。また、BIOS 設定でコンフィデンシャル・コンピューティング機能を有効化する設定が提供されている必要もあります。

15.2 システムの基礎部分の設定

AMD SEV-SNP を有効化した VM ゲスト を動作させるには、まず VM ホストサーバ 側において設定調整を実施する必要があります。SUSE Linux Enterprise Server 15 SP7 では、既定の IOMMU はパススルーに設定されていますが、AMD SEV-SNP を使用するには非パススルーモードに設定しなければなりません。これは、暗号化された VM ゲスト からメモリを介して物理デバイスにアクセスしてしまうと、データの整合性が損なわれる危険性があるためです。なお、任意でインストールする `snphost` ツールを利用するには、MSR カーネルモジュールが必要となります。

1. まずは AMD SEV-SNP のホストツールを提供するパッケージをインストールします:

```
> sudo  zypper install snphost sevctl
```

2. `msr` モジュールを起動時に読み込むように設定するには、下記のように実行します:

```
> sudo  echo "msr" > /etc/modules-load.d/msr.conf
```

3. openSUSE Leap 15.7 で IOMMU の設定を無効化するには、`/etc/default/grub` ファイルを開いて、`GRUB_CMDLINE_LINUX_DEFAULT` 変数内に `iommu=nopt` を追加します。

4. 設定変更後はブートローダの設定を更新します。下記のコマンドを実行してください:

```
> sudo ; update-bootloader
```

5. あとはシステムを再起動することで、Confidential Compute Module が有効化されたカーネルを起動することができます。なお、ブートローダ側の設定で既定のカーネルとして選択されていない場合は、起動メニューで選択しておいてください。

15.3 設定の確認

VM ホストサーバ のインストールと設定が正しくできているかどうかを確認するには、dmesg もしくは任意でインストールする `snphost` ツールを使用します。

- また、カーネルの起動時に AMD Secure Processor の初期化結果が出力されていることを確認します。下記のようにコマンドを実行して確認してください:

```
> sudo  dmesg | grep -i ccp
[ 10.103166] ccp 0000:42:00.1: enabling device (0000 -> 0002)
[ 10.114951] ccp 0000:42:00.1: no command queues available
[ 10.127137] ccp 0000:42:00.1: sev enabled
[ 10.133152] ccp 0000:42:00.1: psp enabled
[ 10.240817] ccp 0000:42:00.1: SEV firmware update successful
```

```
[ 11.128307] ccp 0000:42:00.1: SEV API:1.55 build:8
[ 11.135057] ccp 0000:42:00.1: SEV-SNP API:1.55 build:8
```

上記のように SEV-SNP API のバージョン文字列が表示されていれば、AMD Secure Processor の準備ができたことになります。なお、上記のように表示されていない場合は、BIOS 設定が正しくないか、IOMMU の設定が更新されていないものと考えられます。

15.4 AMD SEV-SNP 仮想マシンのインストール

VM ホストサーバ 側で設定作業を行い、AMD Secure Processor の初期化を行ったあとは、`libvirt` フレームワークで AMD SEV-SNP による保護のある仮想マシンを作成し、インストールすることができます。インストールの手順は通常と同じで、第9章「ゲストのインストール」に示している手順でインストールを行うことができます。ただし、仮想 CD-ROM ドライブによる ISO イメージからのインストールには対応していません。その他の詳しい制限事項については、制限事項の章をお読みください。

なお、仮想 CD-ROM 以外にも、AMD SEV-SNP で保護した仮想マシンの場合、VM ホストサーバが提供する TPM デバイスは利用できません。インストール時の設定に含まれている場合は、これを取り除いてください。このほか、SEV-SNP で保護した仮想マシンは再起動にも対応していません。そのため、インストール中に再起動が必要となる場合は、いったんシャットダウンするように設定してください。その他の詳しい制限事項については、制限事項の章をお読みください。

また 9.1項「GUI ベースのゲストインストール」では、virt-manager を利用した GUI ベースの仮想マシンインストールの手順を説明しています。なお、AMD SEV-SNP を有効化するには、新しい仮想マシンの作成 の最後で 完了 を押す前に、インストールの前に設定をカスタマイズする を選択する必要があることに注意してください。VM ゲスト の設定ダイアログが表示されたら、メモリー を選んで Enable launch security のチェックボックスを選択して 適用 を押します。あとは インストールの開始 を選択するだけで、9.1項「GUI ベースのゲストインストール」で示した手順でインストールを行うことができます。

このほか 9.2項「`virt-install` によるコマンドラインからのインストール」では、`virt-install` による仮想マシンインストールを説明しています。AMD SEV-SNP を有効化したい場合は、`--launchSecurity` オプションを指定してください。下記は 例9.2「`virt-install` コマンドラインの例」でのインストールの際、AMD SEV-SNP による保護を追加した場合の例です:

```
> virt-install --connect qemu:///system --virt-type kvm \
--name sle15sp7 --memory 4096 --disk size=60 --location /path/to/iso --graphics vnc \
--os-variant sle15sp7 --boot uefi --events on_reboot=destroy --launchSecurity type=sev-snp
```

なお、`--launchSecurity` に関する詳細については、https://libvirt.org/formatdomain.html#launch-security にある Domain XML format マニュアル内の Launch Security の章をお読みください。

15.5 AMD SEV-SNP 仮想マシンの確認

仮想マシンの実行状態だけではコンフィデンシャル・コンピューティングが有効化されているかどうかがわかりません。代わりに仮想マシン内で確認する方法がいくつか提供されています。

カーネルのログには、仮想マシン内での AMD メモリ暗号化機能に関するメッセージが出力されます。カーネルのログを確認するには、下記のようなコマンドを実行します:

```
> sudo  dmesg | grep -i sev-snp
[ 1.986186] Memory Encryption Features active: AMD SEV SEV-ES SEV-SNP
```

カーネルログ内にメモリの暗号化機能として SEV-SNP が表示されていれば、コンフィデンシャル・コンピューティングが動作していて、仮想マシンに対してそれが有効化されていることがわかります。

仮想マシン内から SEV-SNP 機能が有効化されているかどうかを確認したい場合は、任意でインストールできる `snpguest` ツールを使用します。`snphost` ツールと同様に、`snpguest` では MSR カーネルモジュールが必要となります。次に仮想マシン内にパッケージをインストールします:

```
> sudo  zypper install snpguest
```

`snpguest` を実行すると、下記のように仮想マシン内からメモリの暗号化機能に関する情報を取得することができます:

```
> sudo  modprobe msr && snpguest ok
[ PASS ] - SEV: ENABLED
[ PASS ] - SEV-ES: ENABLED
[ PASS ] - SNP: ENABLED
```

AMD SEV-SNP 環境下での暗号化の安全性の証明について、詳しくは 15.6項「AMD SEV-SNP 仮想マシンの認証」をお読みください。

15.6 AMD SEV-SNP 仮想マシンの認証

SEV-SNP の有効化が確認できたら、あとは認証処理を行うことで仮想マシンの機密性を確立することができるようになります。この認証処理は暗号処理をベースにした認証で、階層構造による証明関係を構築することで、検証済みのファームウエアや TCB レベルを使用した、正規の AMD ハードウエアであることを証明します。

なお、下記の手順ではローカルでの認証確認のみを実施することに注意してください。ゲスト内で生成されたレポート署名と証明書の連鎖を検証するとともに、認証データがプラットフォームの状態と一致していることを確認します。認証レポートを外部の検証器や検証サービスに送信して、プラットフォームの信頼性を独自に確認し、機密データや処理の認可を行うようなリモート認証は行いません。

認証処理に際しては、`snpguest` と `snphost` というツールを使用します。

15.6.1 認証レポートの生成と検証

ゲスト内で `snpguest` ツールを使用することで、認証ワークフローを実行することができます。この処理では認証レポートを生成し、それぞれの AMD 証明書の連鎖にアクセスして、正しいプラットフォーム鍵で電子的に署名されていることを検証します。

1. まずは認証レポートと対応する要求ファイルを生成します。`--random` フラグを指定することで、唯一性を保証するための乱数データを含めることができます:

```
> sudo snpguest report attestation-report.bin request-file.bin --random
```

2. 次に鍵配布サービス (KDS) から AMD CA と ASK の各証明書を DER 形式で取得します。ここで、`genoa` の箇所にはプロセッサモデルを指定します (モデルが異なる場合は変更してください):

```
> sudo snpguest fetch ca der genoa ./certs-kds
```

3. さらに生成された認証レポートを利用して、Versioned Chip Endorsement Key (VCEK) を取得します:

```
> sudo snpguest fetch vcek der genoa ./certs-kds attestation-report.bin
```

4. 取得した証明書を利用して、認証レポートを検証します:

```
> sudo snpguest verify attestation ./certs-kds attestation-report.bin

Reported TCB Boot Loader from certificate matches the attestation report.
Reported TCB TEE from certificate matches the attestation report.
Reported TCB SNP from certificate matches the attestation report.
Reported TCB Microcode from certificate matches the attestation report.
Chip ID from certificate matches the attestation report.
VEK signed the Attestation Report!
```



注記

なお、`snpguest certificates` コマンドを利用した拡張型の認証ワークフローは QEMU 側の機能に依存して動作するものですが、こちらは現時点では利用できません。

15.6.2 ホスト側での AMD 証明書の検証

ホスト側では、必要に応じてゲスト側の認証レポートを検証するための AMD 証明書連鎖を取得および検証することができます。

1. AMD 社の鍵配布サービス (KDS) から AMD CA と ASK の各証明書を取得します:

```
> sudo  snphost fetch ca pem ./certs
```

2. 次に、プラットフォームに対応した Chip Endorsement Certificate (VCEK または VLEK) を取得します:

```
> sudo  snphost fetch vek pem ./certs
```

3. 取得した証明書の連鎖の正当性を検証します:

```
> sudo  snphost verify ./certs
• = self signed, # = signs, •# = invalid self sign, ## = invalid signs

ARK •
ARK # ASK
ASK # VCEK
```

15.7 現時点での制限事項

SEV-SNP を有効化した VM ゲスト を動作させた場合、下記の制限があることに注意してください。

- SEV-SNP で保護された VM 内で動作させるオペレーティングシステムは、SEV-SNP に対応していなければなりません。SUSE Linux Enterprise Server 15 SP6 またはそれ以降のバージョンであれば、SEV-SNP に対応しています。
- SEV-SNP で保護された VM ゲスト の場合、`virt-install` で `--cdrom` オプションを指定したインストールは実施できません。代わりに `--location` オプションをお使いください。なお、`--location` オプションに関する説明については、`virt-install` のマニュアルページをお読みください。
- SEV-SNP で保護された VM ゲスト では、仮想 CPU 数は最大 255 個までの対応となります。
- SEV-SNP で保護された VM ゲスト は Secure Boot との互換性がありません。Secure Boot サポートが有効化された UEFI ファームウエアを使用すると、SEV-SNP 対応の VM ゲスト が動作しなくなります。
- SEV-SNP で保護された VM ゲスト は、VM ホストサーバ の提供する TPM デバイスを使用できません。これは TPM デバイスを擬似する場合と、パススルーする場合の両方に当てはまります。
- SEV-SNP で保護された VM ゲスト では、ホストデバイスのパススルー (PCI パススルー) を使用することができません。

- SEV-SNP で保護された VM ゲスト では、メモリバルーン機能には対応しません。
- SEV-SNP で保護された VM ゲスト では、メモリや仮想 CPU のホットプラグには対応しません。
- SEV-SNP で保護された VM ゲスト では、huge page 機能には対応しません。
- SEV-SNP で保護された VM ゲスト では、`reboot` や `shutdown -r now` で再起動を行うことができません。再起動はいったんシャットダウンしてから起動し直す方式になります。
- VM ゲスト のメモリ内容の保存と復元 (つまり VM ゲスト の状態保存) には対応していません。言い換えると、SEV-SNP で保護された VM ゲスト ではスナップショット機能が利用できないばかりか、ライブマイグレーションにも対応できません。SEV-SNP を別のホストに移行したい場合は、いったんシャットダウンして移行し、移行先で起動し直す手順が必要になります。

これらの制限事項は様々なハードウエアやファームウェア、およびソフトウエアの特定レイヤが原因となって発生しているもので、将来的に解決できるかもしれません。

16 VM ゲスト の移行

改訂履歴

2025-06-12

仮想化における最大の利点として、VM ゲスト を移動できるという点があります。たとえば VM ホストサーバ でメンテナンスを実施しなければならないような場合や、VM ホストサーバ の負荷が過剰である場合、ゲストを他の VM ホストサーバ に移行することができます。KVM と Xen では「ライブ」マイグレーションにも対応していますので、これにより VM ゲスト を動作させたまま移行することもできます。

16.1 移行の種類

仮想マシン (VM) の移行に際しては、要件に応じて下記の 3 種類の方式から選択することができます。

ライブマイグレーション

移行元の VM を動作させたまま、設定とメモリ内容を移行先のホストに転送します。全ての転送が完了した時点で移行元の VM を一時停止し、移行先の VM を復元します。
ライブマイグレーションは、常に動作させておく必要のある VM を移行したい場合に有用です。



注記

ライブマイグレーションを行う場合、I/O 負荷やメモリページへの書き込みが非常に多くなります。このような場合は、非ライブマイグレーションやオフラインマイグレーションの使用をご検討ください。

非ライブマイグレーション

いったん移行元の VM を一時停止して、設定とメモリを移行先のホストに転送します。転送完了後、移行先で VM を再開します。
非ライブマイグレーションは、ライブマイグレーションより信頼性の高い方式ではありますが、VM にアクセスできなくなる時間が存在するという問題があります。このようなダウンタイムを許容できるシステムであれば、ライブマイグレーションの難しい VM には有用な選択肢となります。

オフラインマイグレーション

VM の設定を移行先のホストにコピーするだけです。移行元の VM は停止させず、移行先の VM を復元させない方式です。
オフラインマイグレーションは、その時点で動作させていない VM を移行する際に使用します。



重要

オフラインマイグレーションを行う場合、`--persistent` オプションを指定しておかなければなりません。

16.2 移行における要件

VM ゲスト を他の VM ホストサーバ に移行する場合は、下記の要件を満たす必要があります:

- 移行元と移行先のシステムが同じアーキテクチャであること。
- NFS や iSCSI など、両方のマシンから同じストレージデバイスにアクセスできること。詳しくは 第 12章「高度なストレージ設定」をお読みください。
  なお、移行の時点で接続済みの CD-ROM やフロッピィディスクのイメージについても、両方のマシンからアクセスできるようにしなければなりません。ただし、これらは移行実施前に取り外しておくことができます。詳しくは 13.11項「仮想マシンマネージャ を利用したフロッピィもしくは CD/DVD-ROM メディアの取り出しと交換」をお読みください。
- 両方の VM ホストサーバ で `libvirtd` を動作させ、かつ移行元と移行先の間でリモートの `libvirt` 接続を許可しておくこと。詳しくは 11.3項「リモート接続の設定」をお読みください。
- 移行先のホストでファイアウオールを動作させている場合は、移行時に使用するポートを開いておく必要があります。移行処理時に特に何も指定しない場合、`libvirt` では 49152:49215 の範囲からポート番号を選択します。移行先のホスト でこのポート範囲の通信を許可するように設定するか、もしくは独自にポートを選択して許可し、移行処理時にそのポート番号を指定してください。
- 移行元と移行先のホストが、ネットワーク内の同じサブネットに属していること。異なるサブネットに属している場合、移行後に接続できなくなってしまいます。
- 移行元と移行先の VM ホストサーバ で、qemu ユーザ, kvm グループ, qemu グループ, libvirt グループの UID や GID が一致していること。
- 移行先のホストで既に同名の VM ゲスト が設定されていないこと。同名の VM ゲスト が存在していて停止している場合、設定は上書きされてしまいます。
- CPU モデルの設定で host cpu モデルを選択していないこと。
- SATA ディスクデバイスを使用していないこと。
- ファイルシステムのパススルー機能を使用していないこと。

- VM ホストサーバ と VM ゲスト の間で適切な時刻維持機能が設定されていること。詳しくは 第19章「VM ゲスト の時刻設定」をお読みください。
- VM ホストサーバ から VM ゲスト に対して、物理デバイスに直接アクセスできる機能を提供していないこと。現時点でのライブマイグレーションは、PCI パススルーや SR-IOV を使用している場合、サポート対象外となります。ライブマイグレーションを使用したい場合は、ソフトウエアによる仮想化 (準仮想化または完全仮想化) をお使いください。
- キャッシュモードを適切に設定していること。詳しくは 18.6項「キャッシュモードとライブマイグレーションの関係」をお読みください。
- 両方のホストでイメージディレクトリが同じパスであること。
- 全てのホストは同一レベルのマイクロコード (特に Spectre マイクロコード更新) が適用されていること。これは全てのホストで openSUSE Leap の最新版をインストールすることで実現できます。

16.3 仮想マシンマネージャ によるライブマイグレーション

仮想マシンマネージャ で VM ゲスト の移行を行う場合、どちらのマシンで処理を開始してもかまいません。移行元のホストで 仮想マシンマネージャ を実行してもかまいませんし、全く別のホストで実行してもかまいません。ただし、後者の場合は移行元と移行先の両方に接続できるように設定しておく必要があります。

1. まずは 仮想マシンマネージャ を起動し、移行元と移行先に接続します。移行元のホストでも移行先のホストでもない場所から 仮想マシンマネージャ を起動している場合は、両方のホストに接続してください。
2. 移行したい VM ゲスト を選択して右クリックし、［移行］を選択します。なお、対象のゲストが実行中か、もしくは一時停止されている必要があります。停止中の場合は移行できません。

   ### ヒント: 移行処理の高速化について

   移行処理を高速化するためには、VM ゲスト を一時停止するのが最適です。これは 16.1項「移行の種類」で説明している「非ライブマイグレーション」と同じ処理になります。

3. VM ゲスト の移行先となる［新しいホスト］を選択します。移行先が表示されない場合は、移行先に対して接続を実施しているかどうかをご確認ください。

移行先のホストに対する接続オプションを変更したい場合は、［接続］内の［モード］,［アドレス］(IP アドレスまたはホスト名),［ポート］をそれぞれ設定します。なお、［ポート］を指定した場合は［アドレス］の設定も行わなければなりません。
また、［詳細なオプション］内では、恒久的な移行か一時的な移行かを選択することができます。一時的に移行するだけであれば、［一時的に移動］を選択します。
このほか、［安全ではない場合を許可する］というオプションも用意されています。これは VM ホストサーバ のキャッシュ機構を無効化することなく移行を実施するためのオプションで、移行処理を高速化することができますが、`cache="none"` / `O_DIRECT` を使用しておらず、VM ゲスト のストレージに対する一貫したビューが利用可能な場合にのみ動作します。



注記: 帯域オプションについて

仮想マシンマネージャ の新しいバージョンでは、移行時の帯域設定に関する設定オプションが削除されています。帯域を指定して移行を行いたい場合は、`virsh` をお使いください。

4. 移行を開始するには［マイグレーション］を押します。
移行処理が完了すると［マイグレーション］ウインドウが閉じ、仮想マシンマネージャ ウインドウ内の移行先ホスト内に VM ゲスト が表示されるようになります。なお、移行元の VM ゲスト についても、シャットダウン状態で残ります。

16.4 `virsh` による移行

VM ゲスト を `virsh migrate` コマンドで移行するには、VM ホストサーバ に対する直接のアクセスまたはリモートからのシェルアクセスが必要となります。これは、コマンドをホスト内で実行する必要があるためです。移行のコマンドは下記のように記述します:

```
> virsh migrate [オプション] VM_ID_または名前 接続_URI [--migrateuri tcp://リモートホスト:ポート]
```

下記に主要なオプションを示します。完全な一覧を読みたい場合は、`virsh help migrate` コマンドを実行してください。

`--live`
: ライブマイグレーションを実行します。このオプションを指定しない場合、ゲストは移行の際に一時停止します (「非ライブマイグレーション」になります) 。

`--suspend`
: ライブマイグレーションや非ライブマイグレーションで、移行先のホストを一時停止したままの状態にします。

`--persistent`

移行先のホストで恒久的な VM 移行を実施します。このオプションを指定しないと、VM をシャットダウンした段階で `virsh list --all` の一覧には表示されなくなります。

`--undefinesource`

このオプションを指定すると、移行が成功した時点で移行元のホストでの VM ゲスト の定義を削除します。なお、ゲストに接続されている仮想ディスクは削除されない ことに注意してください。

`--parallel --parallel-connections` 接続数

パラレルマイグレーション (並行移行) は、単一スレッドの移行処理ではネットワークの帯域を使い切れないような場合に使用するもので、移行処理をより高速化する目的で使用されます。たとえば 40 GB のネットワークインターフェイスを持つホストの場合、スレッドを 4 つにすることでネットワーク帯域を埋めることができるようになります。またパラレルマイグレーションを使用することで、移行処理に必要な巨大なメモリ占有の時間を減らすこともできます。

下記の例では、移行元を mercury.example.com とし、移行先を jupiter.example.com とした移行処理を実行しています。また、VM ゲスト の名前は `opensuse131` で、ID が `37` である場合の例となります。

既定のパラメータを利用した非ライブマイグレーションの例

```
> virsh migrate 37 qemu+ssh://tux@jupiter.example.com/system
```

既定のパラメータを利用した一時的なライブマイグレーションの例

```
> virsh migrate --live opensuse131 qemu+ssh://tux@jupiter.example.com/system
```

恒久的なライブマイグレーションで、移行元の VM 定義を削除する例

```
> virsh migrate --live --persistent --undefinesource 37 \
qemu+tls://tux@jupiter.example.com/system
```

ポート 49152 を利用した非ライブマイグレーション

```
> virsh migrate opensuse131 qemu+ssh://tux@jupiter.example.com/system \
--migrateuri tcp://@jupiter.example.com:49152
```

使用している全てのストレージを転送するライブマイグレーション

```
> virsh migrate --live --persistent --copy-storage-all \
opensuse156 qemu+ssh://tux@jupiter.example.com/system
```



重要

`--copy-storage-all` オプションを指定して VM のストレージを転送する場合、ストレージは `libvirt` のストレージプール内に存在していなければなりません。また、移行先でも同じ種類かつ同じ名前のストレージプールが存在していなければなりません。

移行元でストレージプールの XML 形式出力を行いたい場合は、下記のようなコマンドを入力して実行します:

```
> sudo virsh pool-dumpxml VM_名 > EXAMPLE_POOL.xml
```

移行先のホストでストレージプールを作成して開始するには、上記で出力した XML ファイルをコピーしてから下記のコマンドを実行します:

```
> sudo virsh pool-define EXAMPLE_POOL.xml
> sudo virsh pool-start EXAMPLE_VM
```



注記: 一時的な移行と恒久的な移行

既定では、`virsh migrate` は移行先のホスト内に一時的な VM ゲスト のコピーを作成します。移行元のホストには元のゲストの設定が残り、ゲストはシャットダウン状態になります。また、移行先のゲストをシャットダウンすると、一時的な移行の場合は削除されてしまいます。

移行先のホストで恒久的に動作させたい場合は、`--persistent` オプションを指定してください。この場合も移行元のホストには、ゲストがシャットダウンした状態で残されます。`--persistent` に加えて `--undefinesource` を指定することで、移行先のホストに恒久的な移行を実施し、移行元のゲストを削除するようになります。

なお、`--persistent` オプションを指定せずに `--undefinesource` オプションを指定してしまうと、移行先のホストではゲストをシャットダウンしたタイミングで設定が失われてしまうため、両方のホストからゲストが削除されてしまうことに注意してください。

16.5 手順例

16.5.1 ストレージのエクスポート

まずはホスト間でゲストイメージを共有するため、ストレージのエクスポート (公開) を実施します。一般的には NFS サーバを利用します。下記の例では、`/volume1/VM` ディレクトリを 10.0.1.0/24 のネットワーク内にある全てのホストに対して提供します。具体的には、root ユーザで `/etc/exports` ファイルを編集して、下記の内容を追記します:

```
/volume1/VM 10.0.1.0/24  (rw,sync,no_root_squash)
```

設定変更後は NFS サーバの再起動が必要です:

```
> sudo systemctl restart nfsserver
> sudo exportfs
/volume1/VM      10.0.1.0/24
```

16.5.2 移行先ホストでのプールの設定

VM ゲスト の移行を行いたい全てのホストにおいて、ボリュームへのアクセスを許可するためのプール定義を実施しなければなりません。ここでは NFS サーバのアドレスが 10.0.1.99 であり、`/volume1/VM` ディレクトリを `/var/lib/libvirt/images/VM` ディレクトリにマウントするものとします。また、プール名は VM であるものとします。プールを定義するには、`VM.xml` ファイルを作成して下記のような内容を記述します:

```
<pool type='netfs'>
  <name>VM</name>
  <source>
    <host name='10.0.1.99'/>
    <dir path='/volume1/VM'/>
    <format type='auto'/>
  </source>
  <target>
    <path>/var/lib/libvirt/images/VM</path>
    <permissions>
      <mode>0755</mode>
      <owner>-1</owner>
      <group>-1</group>
    </permissions>
  </target>
  </pool>
```

あとは `pool-define` コマンドを利用して `libvirt` に読み込みます:

```
# virsh pool-define VM.xml
```

このほかにも、プールの定義は `virsh` コマンドで直接実施することもできます:

```
# virsh pool-define-as VM --type netfs --source-host 10.0.1.99 \
     --source-path /volume1/VM --target /var/lib/libvirt/images/VM
プール VM が作成されました
```

下記のコマンドは、`virsh` に何もパラメータを付与せずに実行した場合に起動される、`virsh` の対話型シェル内で実行した場合の例となります。あとはホストの起動時に自動的にプールを開始するように設定します (autostart オプションを指定します):

```
virsh # pool-autostart VM
プール VM が自動起動としてマークされました
```

自動起動を無効化したい場合は、下記のように入力して実行します:

```
virsh # pool-autostart VM --disable
プール VM の自動起動マークが解除されました
```

プールが存在しているかどうかを確認します:

```
virsh # pool-list --all
 名前                 状態       自動起動
------------------------------------------
 default              動作中     はい (yes)
 VM                   動作中     はい (yes)

virsh # pool-info VM
名前:            VM
UUID:            42efe1b3-7eaa-4e24-a06a-ba7c9ee29741
状態:            実行中
永続:            はい (yes)
自動起動:        はい (yes)
容量:            2,68 TiB
割り当て:        2,38 TiB
利用可能:        306,05 GiB
```



警告: 全ての宛先ホストにプールが必要となる件について

注意: VM ゲスト の移行を行う場合、移行元と移行先の両方で同じプールを定義しなければなりません。

16.5.3 ボリュームの作成

プールを定義したら、次はディスクイメージを保持するボリュームを定義します:

```
virsh # vol-create-as VM sled12.qcow2 8G --format qcow2
ボリューム sled12.qcow2 が作成されました
```

ここで設定したボリューム名は、virt-install でゲストをインストールする際に使用します。

16.5.4 VM ゲスト の作成

あとは `virt-install` コマンドで openSUSE Leap の VM ゲスト を作成するだけです。VM プールは `--disk` オプションで指定しますが、移行時に `--unsafe` オプションを指定しなくて済むようにするため、cache=none を指定しておくことをお勧めします。

```
# virt-install --connect qemu:///system --virt-type kvm --name \
   sles15 --memory 1024 --disk vol=VM/sled12.qcow2,cache=none --cdrom \
   /mnt/install/ISO/SLE-15-Server-DVD-x86_64-Build0327-Media1.iso --graphics \
   vnc --os-variant sled15
インストールの開始中...
ドメインを作成中...
```

16.5.5 VM ゲスト の移行

これで移行に関する全ての準備ができました。あとは現時点で VM ゲスト が動作している VM ホスト サーバ で、移行先を指定して `migrate` コマンドを実行するだけです。

```
virsh # migrate --live sled12 --verbose qemu+ssh://IP/ホスト名/system
パスワード:
Migration: [ 12 %]
```

17 Xen から KVM への移行ガイド

改訂履歴

2025-06-12

KVM の仮想化ソリューションはサーバ管理者の間でも非常に知られた存在となっており、既存の Xen ベースの環境を KVM に移行する要望も多くなっています。ですが、現時点では Xen の VM を KVM に自動変換してくれるような成熟したツールは提供されていません。その代わり、Xen の仮想マシンを KVM に移行する作業を支援する技術ソリューションは用意されています。下記の章では、このような移行を行うための情報と手順を説明しています。

重要: 移行手順はサポート対象外である件について

この文書内で説明している移行手順は、SUSE では完全にはサポートしていません。ガイダンスとしてのみ提供しているものです。

17.1 `virt-v2v` を使用した KVM への移行

本章では、KVM 以外のハイパーバイザ (たとえば Xen) から `libvirt` が管理する KVM への仮想マシンの取り込み方法について説明しています。

ヒント: Microsoft Windows のゲストについて

本章では Linux ゲストに主眼を置いて説明しています。`virt-v2v` を利用して Microsoft Windows のゲストを移行する手順は Linux ゲストの場合とほとんど同じですが、仮想マシンドライバパック (VMDP) の部分のみ違いがあります。VMDP を利用した Windows ゲストの変換方法について、詳しくは Virtual Machine Driver Pack documentation (https://documentation.suse.com/sle-vmdp/) をご覧ください。

17.1.1 `virt-v2v` の紹介

`virt-v2v` は KVM 以外のハイパーバイザで動作する VM ゲストを、`libvirt` が管理する KVM 上で実行できるようにするコマンドラインツールです。可能であれば、変換後の仮想マシンで準仮想化型の virtio ドライバを使用するように設定することもできます。サポート対象となるオペレーティングシステムと、ハイパーバイザの一覧は下記のとおりです:

サポート対象のゲスト側オペレーティングシステム

- SUSE Linux Enterprise Server
- openSUSE
- Red Hat Enterprise Linux
- Fedora
- Microsoft Windows Server 2003 または 2008

サポート対象の移行元ハイパーバイザ

- Xen

サポート対象の移行先ハイパーバイザ

- KVM (`libvirt` での管理となります)

17.1.2 `virt-v2v` のインストール

`virt-v2v` のインストールは簡単です:

```
> sudo zypper install virt-v2v
```

なお、`virt-v2v` コマンドを実行するには `root` の権限が必要となります。`root` で実行するか、もしくは `sudo` を介して実行してください。

17.1.3 `libvirt` 管理下の KVM 仮想マシンへの変換

`virt-v2v` は、Xen ハイパーバイザで動作している仮想マシンを `libvirt` 管理下の KVM に変換することができるツールです。`libvirt` や `virsh` コマンドに関する詳細については、パート II「libvirt を利用した仮想マシンの管理」をお読みください。そのほか、`virt-v2v` のコマンドラインオプションについては、`virt-v2v` のマニュアルページ (`man 1 virt-v2v`) をお読みください。

仮想マシンを変換する前に、まずは下記の手順を実行しておきます:

手順 17.1: 移行のための環境準備

1. 新しいローカルストレージプールを作成します。
   `virt-v2v` は、移行元の仮想マシンのストレージを `libvirt` の管理下にあるローカルのストレージプールにコピーします (移行元のディスクイメージはそのまま保持されます) 。プールを作成するには 仮想マシンマネージャ を使用するか、もしくは `virsh` を使用してください。詳しくは 8.2.2項「仮想マシンマネージャ を利用したストレージの管理」および 8.2.1項「`virsh` を利用したストレージの管理」をお読みください。

2. ローカルのネットワークインターフェイスを準備します。
   変換後の仮想マシンが VM ホストサーバ のローカルネットワークインターフェイスを使用できるかどうかを確認します。通常はネットワークブリッジを使用しますが、ネットワークブリッジを作成していない場合は、[YaST] › [システム] › [ネットワークの設定] › [追加] › [ブリッジ] で作成してください。



注記: ネットワークデバイスのマッピングについて

移行元で Xen ホスト内のネットワークデバイスを使用している場合は、変換処理内で移行先の KVM ホスト内のネットワークデバイスを使用するようにすることができます。たとえば Xen のネットワークブリッジが `br0` であれば、そのまま KVM でも使用することができます。`/etc/virt-v2v.conf` ファイル内では、そのような変換ルールを設定することができます。この変換機能を有効化するには、`<!--` と `-->` でコメントアウトされた箇所のコメントを外してください。たとえば下記のようになります:

```
<network type='bridge' name='br0'>
  <network type='network' name='default'/>
</network>
```



ヒント: ネットワークブリッジが無い場合について

ネットワークブリッジが無い場合、必要であれば 仮想マシンマネージャ で作成することもできます。

`virt-v2v` は下記のような書式で実行することができます:

```
virt-v2v -i 入力方式 -os ストレージプール 移行元_VM
```

入力方式
: 入力方式を指定します。`libvirt` または `libvirtxml` のいずれかを指定してください。詳しくは `移行元_VM` の説明をお読みください。

ストレージプール
: 移行先の仮想マシン向けに用意したストレージプールを指定します。

移行元_VM
: 移行元の仮想マシンを指定します。ここで指定すべき値は `入力方式` の指定によって異なります。`libvirt` を指定した場合は移行元のドメイン名を、`libvirtxml` を指定した場合は移行元のドメインの XML 設定ファイルのパスを指定します。



注記: 変換にかかる時間について

仮想マシンの変換には、主にディスクイメージ全体をコピーする処理として、多くのシステム資源が必要となります。1 台の仮想マシンあたり最大でも 10 分程度が必要になりますが、ディスクイメージが大きい場合はより多く時間がかかる場合もあります。

17.1.3.1 `libvirt` の XML 設定ファイルをベースにした変換

本章では、`libvirt` の XML 設定ファイルを利用してローカルの Xen 仮想マシンを変換する方法について説明しています。この方式は、既に KVM ハイパーバイザが動作している環境に適切な仕組みです。なお、移行元の `libvirt` の XML ファイルと `libvirt` のストレージプールが、それぞれローカルホスト内に存在していることを確認しておいてください。

1. まずは移行元の仮想マシンに対する `libvirt` の XML 設定ファイルを取得します。



ヒント: XML ファイルの取得について

移行元の仮想マシンの `libvirt` XML ファイルを取得するには、まず Xen カーネルを利用してホスト側の OS を動作させなければなりません。既に KVM が有効化された環境を起動している場合は、まず Xen カーネルに戻してから `libvirt` の XML ファイルを取得し、その後再度 KVM 環境にしてください。

まずは移行元の仮想マシンを virsh で識別します:

```
# virsh list
 Id    名前                           状態
```

```
----------------------------------------------------
[...]
  2     sles12_xen                        実行中
[...]
```

変換元の仮想マシンが `sles12_xen` である場合は、これを XML 形式のファイル `sles12_xen.xml` に保存するには、下記のようなコマンドを実行します:

```
# virsh dumpxml sles12_xen > sles12_xen.xml
```

2. 出力された XML ファイルの内容を表示させて、KVM ホストの観点からもアクセス可能なパスであることを確認してください。1 台のマシン内でローカル移行するだけであれば問題はありませんが、他のホストに移行させるような場合は、パスを変更する必要があるかもしれません。

```
<source file='/var/lib/libvirt/images/XenPool/SLES.qcow2'/>
```



ヒント: イメージのコピーについて

イメージを何度もコピーしてしまわないようにするため、ディスクイメージを `libvirt` のストレージプールに直接コピーしておくことをお勧めします。この場合は、XML ファイル内の source file の内容を合わせて変更する必要があります。なお、`virt-v2v` を使用すると、既存のディスクを検出してその場で変換を行うようになります。

3. `virt-v2v` コマンドを実行して KVM 仮想マシンに変換します:

```
# virt-v2v sles12_xen.xml❶ \
-i XML_ファイル❷ \
-os 移行先のホスト:/エクスポートされたディレクトリ❸ \
--bridge br0❹ \
-on sles12_kvm❺
```

❶ 移行元の Xen ベースの仮想マシンの XML ファイルを指定します。

❷ `virt-v2v` は移行元の仮想マシンに関する情報を、`libvirt` の XML ファイルから読み込みます。

❸ 移行先の仮想マシンのディスクイメージを配置するストレージプールを指定します。この例では、`移行先のホスト` 内にある `/エクスポートされたディレクトリ` 内にイメージを配置することになります。

❹ 移行先の KVM ベースの仮想マシンで使用するネットワークブリッジを指定します。ここではホスト内の `br0` を使用する意味になります。

❺ 移行先の仮想マシンを `sles12_kvm` という名前に変更して、名前が重複しないように指定しています。

17.1.3.2 libvirt のドメイン名をベースにした変換

この方式は既に `libvirt` で Xen を使用していて、移行後に同じマシンを KVM ハイパーバイザに移行するような場合に有用です。

1. まずは移行対象の仮想マシンの `libvirt` ドメイン名を確認します。

```
# virsh list
 Id    名前                           状態
----------------------------------------------------
[...]
  2     sles12_xen                     実行中
[...]
```

   上記の出力から、`sles12_xen` が移行対象であるものとします。

2. `virt-v2v` コマンドを実行して KVM 仮想マシンに変換します:

```
# virt-v2v sles12_xen❶ \
-i libvirt❷ \
-os ストレージプール❸ \
--network eth0❹ \
-of qcow2❺ \
-oa sparse❻ \
-on sles12_kvm
```

   ❶ Xen ベースの仮想マシンのドメイン名を指定します。

   ❷ `virt-v2v` は `libvirt` との接続を介して仮想マシンの情報を直接取得します。

   ❸ 移行先のディスクイメージをローカルの `libvirt` ストレージプールに配置します。

   ❹ 全てのゲストブリッジ (もしくはネットワーク) は、ローカルで管理するネットワークに接続します。

   ❺ 移行先の仮想マシンで使用するディスクイメージの形式を指定します。`raw` もしくは `qcow2` のいずれかがサポートされています。

   ❻ 変換後のディスク領域の割り当て方法を指定します。`sparse` (スパースファイルとして割り当てる) もしくは `preallocated` (事前に容量を割り当てる) のいずれかを指定します。

17.1.3.3 リモートの Xen 仮想マシンの変換

この方式は、移行元となる Xen の仮想マシンがリモートに存在するような場合に有用です。`virt-v2v` ではリモートのホストとの間を `ssh` 経由で接続しますので、移行元のホストで SSH サービスが動作していることを確認しておいてください。



注記: パスワード無しの SSH アクセスについて

`virt-v2v` を動作させるにはパスワード無しの SSH 接続が必要となります。これはつまり、ssh-agent を利用して SSH 鍵を追加しておく必要があることを意味します。詳しくは `man ssh-keygen` および `man ssh-add` でそれぞれ表示されるマニュアルページをお読みください。『セキュリティ強化ガイド』、第23章「OpenSSH によるネットワーク操作の機密保持」にも詳しい説明があります。

リモートのホストとの間で `libvirt` 接続を行うには、まずリモート側のホストを表す URI を組み立てる必要があります。下記の例は、接続先のホスト名が `remote_host.example.com` で、接続に使用するユーザ名が `root` である場合の例です:

```
xen+ssh://root@remote_host.example.com/
```

`libvirt` の接続 URI に関する詳細は、https://libvirt.org/uri.html をお読みください。

1. 出力された内容から、移行対象の仮想マシンの `libvirt` ドメイン名を探します。

   ```
   # virsh -c xen+ssh://root@remote_host.example.com/ list
    Id    名前                           状態
   ----------------------------------------------------
     1     sles12_xen                     実行中
   [...]
   ```

   上記の出力から、`sles12_xen` が移行対象であるものとします。

2. リモート接続で `virt-v2v` コマンドを使用する場合、下記のようなコマンドラインになります:

   ```
   # virt-v2v sles12_xen \
   -i libvirt \
   -ic xen+ssh://root@remote_host.example.com/ \
   -os ローカルストレージプール \
   --bridge br0
   ```

17.1.4 変換した仮想マシンの起動

`virt-v2v` が完了すると、`-on` オプションで指定した名前で `libvirt` 内に新しいドメインが作成されます。`-on` を指定しない場合、移行元と同じ名前の仮想マシンになります。新しいゲストは標準的な `libvirt` ツール、たとえば `virsh` や 仮想マシンマネージャ などで管理することができます。



ヒント: マシンの再起動について

17.1.3.2項「libvirt のドメイン名をベースにした変換」の手順で Xen の仮想マシンを変換したあとは、ホストマシンを再起動して非 Xen カーネルを起動することができます。

17.2 Xen から KVM への手動移行

17.2.1 概要

仮想マシンを管理する際の推奨される方式は `libvirt` です (詳しくは https://libvirt.org/ をお読みください) 。`libvirt` は仮想マシンを手作業で作成して動作させるよりも手間のかからない方式で、クロスプラットフォームに対応するほか、多数のハイパーバイザにも対応しています。そのほか、リモート接続に際しても機密を保持することができるほか、仮想ネットワークにも対応し、仮想マシンを管理する際の抽象化レイヤとしても使用することができます。そのため、この記事での説明は `libvirt` の方式をベースにしています。

一般的に、Xen から KVM への移行は下記の手順で行います:

1. 移行元の Xen VM ゲスト に対するバックアップコピーを作成する。
2. (任意) 準仮想化ゲストに固有の変更を適用する。
3. 移行元の Xen VM ゲスト に関する情報を取得し、KVM 環境で等価な設定を作成する。
4. Xen ホスト内の移行元ゲストをシャットダウンし、KVM ハイパーバイザ内で移行先のゲストを起動する。



警告: ライブマイグレーションに対応しない件について

Xen から KVM への移行はライブマイグレーションに対応していません。そのため、VM ゲストを動作させたままでは移行を行うことができません。移行先の KVM の VM ゲスト を動作させる前に、移行元の Xen の VM ゲスト をシャットダウンしてください。

17.2.2 Xen VM ゲスト のバックアップ

お使いの Xen VM ゲスト をバックアップするには、下記の手順を実施します:

1. まずは移行元の Xen ゲストを識別します。下記のようにして ID と名前 (Name) を記憶しておきます。

```
> sudo virsh list --all
Id 名前                 状態
----------------------------------
 0 Domain-0             実行中
 1 SLES15SP3            実行中
[...]
```

2. ゲストをシャットダウンします。シャットダウンはゲスト OS 内から実行することが できるほか、下記のようにして `virsh` を利用しても実行できます:

```
> sudo virsh shutdown SLES11SP3
```

3. まずは設定を XML ファイルにバックアップします。

```
> sudo virsh dumpxml SLES11SP3 > sles11sp3.xml
```

4. 次にディスクイメージファイルをバックアップします。`cp` や `rsync` などのコマンドを利用してバックアップコピーを作成してください。このとき、`md5sum` コマンドでコピーのチェックサムを作成しておくとよいでしょう。

5. ディスクイメージファイルをバックアップしたら、再度ゲストを起動します:

```
> sudo virsh start SLES11SP3
```

17.2.3 準仮想化ゲストに固有の変更

準仮想化 Xen ゲストから移行する場合は、下記のような変更を適用します。下記の変更はゲストを動作させて適用してもかまいませんし、`guestfs-tools` を使用すれば停止済みのゲストでも適用することができます。



重要

本章で示した変更を適用してしまうと、移行元の VM ゲスト は Xen 内で動作しなくなってしまいますのでご注意ください。

17.2.3.1 既定のカーネルのインストール



警告: 起動してはならない件について

既定のカーネルをインストールした後は Xen からゲストを起動しようとしないでください。起動しようとしても動作しないためです。

Xen ゲストのディスクイメージを KVM ハイパーバイザ内で使用するためにコピーする場合、Xen ハイパーバイザを 使用せずに 起動できるものであることを確認してください。準仮想化 Xen ゲストの場合、通常は Xen 向けの特殊なカーネルが動作しているほか、GRUB 2 などのブートローダもインストールされていないためです。

1. SLES 11 の場合は `/etc/sysconfig/kernel` ファイルを更新します。`INITRD_MODULES` パラメータ内にある全ての Xen ドライバを削除し、virtio ドライバに置き換えます。たとえば下記前者のような設定になっている場合:

   ```
   INITRD_MODULES="xenblk xennet"
   ```

   上記を下記のように変更します:

   ```
   INITRD_MODULES="virtio_blk virtio_pci virtio_net virtio_balloon"
   ```

   SLES 12, 15 もしくは openSUSE の場合、`/etc/dracut.conf.d/*.conf` ファイル内を検索し、`xenblk xennet` を `virtio_blk virtio_pci virtio_net virtio_balloon` に置き換えます。

2. 準仮想化 Xen ゲストの場合、固有の Xen カーネルを動作させています。KVM で動作できるようにするには、既定のカーネルをインストールする必要があります。



注記: 既に標準のカーネルがインストールされている場合がある件について

完全仮想化ゲストの場合は既に標準のカーネルがインストールされていますので、ここで敢えてインストールし直す必要はありません。

`rpm -q kernel-default` コマンドを Xen ゲスト内で実行すると、既定のカーネルがインストールされているかどうかを確認することができます。インストールされていない場合は `zypper in kernel-default` コマンドでインストールしてください。

なお、KVM でゲストを動作させるには、virtio (準仮想化) ドライバをインストールしておかなければなりません。下記のようなコマンドを実行して確認してください。なお、`6.4.0-150600.9` の箇所をお使いのカーネルバージョンに置き換えて実行してください:

```
> sudo sudo find /lib/modules/6.4.0-150600.9-default/kernel/drivers/ -name virtio*
/lib/modules/6.4.0-150600.9-default/kernel/drivers/block/virtio_blk.ko.zst
/lib/modules/6.4.0-150600.9-default/kernel/drivers/bluetooth/virtio_bt.ko.zst
/lib/modules/6.4.0-150600.9-default/kernel/drivers/char/hw_random/virtio-rng.ko.zst
/lib/modules/6.4.0-150600.9-default/kernel/drivers/crypto/virtio
/lib/modules/6.4.0-150600.9/kernel/drivers/block/virtio_blk.ko
...
```

3. 次に `/etc/fstab` ファイルを更新します。`xvda` デバイスを `vda` デバイスに変更します。

4. さらにブートローダの設定を更新します。まずは Xen ゲスト内で `rpm -q grub2` を実行して、GRUB 2 がインストールされているかどうかを確認します。インストールされていない場合は `zypper in grub2` を実行してインストールしてください。
   OS の起動時に新しくインストールした既定のカーネルを使用するように設定します。なお、Xen 固有のデバイスを使用している場合は、対応するカーネルのコマンドラインオプションを削除もしくは更新してください。この作業は YaST ([システム] › [ブートローダ]) のほか、手作業でも行うことができます:

   - まずはメニュー内の Linux の起動項目のうち、変更したい箇所を識別します:

     ```
     > cat /boot/grub2/grub.cfg | grep 'menuentry '
     ```

     新しくインストールしたものがどれなのかを番号で覚えておきます (上から順に 0 から数えてください) 。

   - 新しくインストールしたカーネルを既定の起動項目に設定します:

     ```
     > sudo grub2-set-default N
     ```

     ここで、`N` の箇所には新しくインストールしたカーネルの順序番号を指定します。

   - `/etc/default/grub` ファイルをエディタなどで開き、`GRUB_CMDLINE_LINUX_DEFAULT` および `GRUB_CMDLINE_LINUX_RECOVERY` のオプションを探します。それらの値のうち、Xen 固有のデバイス指定があれば、それらを削除もしくは更新してください。たとえば下記前者のような指定があった場合:

     ```
     root=/dev/xvda1 disk=/dev/xvda console=xvc
     ```

上記を下記のように変更します:

```
root=/dev/vda1 disk=/dev/vda
```

なお、`xvc` のようなコンソール指定があった場合は、それら全て (`xvc0` など) を削除する必要があります。

5. `/boot/grub2` もしくは `/boot/grub2-efi` ディレクトリ内にある `device.map` ファイルを更新します。ここでは `xvda` のストレージデバイスを `vda` に変更してください。

6. 新しい既定値を取り込むため、下記のように実行します:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

17.2.3.2 KVM で起動できるようにするためのゲストの更新

1. さらにシステムを更新して、既定のシリアルコンソールを使用するようにします。まずは設定済みのコンソールを一覧表示して、`xvc?` へのシンボリックリンクを削除してください。

```
> sudo ls -l /etc/systemd/system/getty.target.wants/
getty@tty1.service -> /usr/lib/systemd/system/getty@.service
getty@xvc0.service -> /usr/lib/systemd/system/getty@xvc0.service
getty@xvc1.service -> /usr/lib/systemd/system/getty@xvc1.service

# rm /etc/systemd/system/getty.target.wants/getty@xvc?.service
```

2. さらに `/etc/securetty` ファイルを更新します。`xvc0` を `ttyS0` に置き換えてください。

17.2.4 Xen VM ゲスト 設定の更新

本章では移行元の Xen VM ゲスト の設定をエクスポートする方法、および `libvirt` の管理下の KVM ゲストとして取り込む際の変更すべき項目について説明しています。

17.2.4.1 Xen VM ゲスト 設定のエクスポート

まずはゲストの設定をエクスポートしてファイルに保存します。保存された内容は、たとえば下記のようになります:

```
> sudo virsh dumpxml SLES11SP3
<domain type='xen'>
  <name>SLES11SP3</name>
```

```
  <uuid>fa9ea4d7-8f95-30c0-bce9-9e58ffcabeb2</uuid>
  <memory>524288</memory>
  <currentMemory>524288</currentMemory>
  <vcpu>1</vcpu>
  <bootloader>/usr/bin/pygrub</bootloader>
  <os>
    <type>linux</type>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/lib/xen/bin/qemu-dm</emulator>
    <disk type='file' device='disk'>
      <driver name='file'/>
      <source file='/var/lib/libvirt/images/SLES_11_SP2_JeOS.x86_64-0.0.2_para.raw'/>
      <target dev='xvda' bus='xen'/>
    </disk>
    <interface type='bridge'>
      <mac address='00:16:3e:2d:91:c3'/>
      <source bridge='br0'/>
      <script path='vif-bridge'/>
    </interface>
    <console type='pty'>
      <target type='xen' port='0'/>
    </console>
    <input type='mouse' bus='xen'/>
    <graphics type='vnc' port='-1' autoport='yes' keymap='en-us'/>
  </devices>
</domain>
```

libvirt での VM ゲスト の XML 書式に関する詳細は、https://libvirt.org/formatdomain.html で説明されています。

17.2.4.2 ゲストの設定に対する一般的な変更

出力した Xen ゲストの XML 設定ファイルは、KVM ハイパーバイザで動作させる際にいくつか修正する必要があります。下記の手順は完全仮想化と準仮想化の両方に対応しています。ただし、下記の XML 要素全てが存在するとは限りません。存在しているもののみを修正してください。



ヒント: 使用する用語について

XML 設定ファイル内での項目位置を示すため、本文書では XPath 文法を使用しています。たとえば下記のような XML ファイル内で、`<domain>` タグ内の `<name>` を表す場合:

```
<domain>
```

```
  <name>sles11sp3</name>
</domain>
```

XPath では `/domain/name` と表記します。

1. まずは `/domain` 要素内の `type` 属性を、`xen` から `kvm` に変更します。
2. `/domain/bootloader` 要素を削除します。
3. `/domain/bootloader_args` 要素を削除します。
4. `/domain/os/type` 要素の値を、`linux` から `hvm` に変更します。
5. `/domain/os` 要素内に `<boot dev="hd"/>` を追加します。
6. `/domain/os/type` 要素内に `arch` 属性を追加します。設定可能な値は `arch="x86_64"` もしくは `arch="i686"` のいずれかです。
7. `/domain/devices/emulator` 要素内の値を、`/usr/lib/xen/bin/qemu-dm'` から `/usr/bin/qemu-kvm` に変更します。
8. 準仮想化 (PV) ゲストに関連づけられたディスクに対して、下記の設定変更を行います:
   - `/domain/devices/disk/driver` 要素内にある `name` 属性を `file` から `qemu` に変更し、`type` 属性を追加します。`type` 属性には `raw` もしくは `qcow2` のいずれかの値を指定します。
   - `/domain/devices/disk/target` 要素内の `dev` 属性を、`xvda` から `vda` に変更します。
   - `/domain/devices/disk/target` 要素内の `bus` 属性を、`xen` から `virtio` に変更します。

9. それぞれのネットワークカードに対して、下記の変更を行います:

   - `/domain/devices/interface` 内に `model` 要素が存在している場合は、その `type` 属性を `virtio` に変更します:

     ```
     <model type="virtio">
     ```

   - `/domain/devices/interface/script` 要素があれば、それらの要素全てを削除します。

   - `/domain/devices/interface/target` 要素内に `dev` 属性が存在していて、その値が `vif`, `vnet`, `veth` のいずれかで始まるものであった場合は、それらの要素全てを削除します。独自のネットワークを使用している場合は、`dev` の値を変更します。

10. `/domain/devices/console` 要素があれば、それらの要素全てを削除します。

11. `/domain/devices/serial` 要素があれば、それらの要素全てを削除します。

12. `/domain/devices/input` 要素内の `bus` 属性を、`xen` から `ps2` に変更します。

13. メモリバルーン機能を使用する場合は、`/domain/devices` 要素内に下記の要素を追加します。

    ```
    <memballoon model="virtio"/>
    ```



ヒント: デバイス名について

`<target dev='hda' bus='ide'/>` 要素はゲスト OS 側に対してどのようにディスクを提示するのかを制御する項目です。`dev` 属性では「論理」デバイス名を指定しますが、この値とゲスト OS 内でのデバイス名は、必ずしも一致しているとは限りません。そのため、ブートローダのコマンドラインで、ディスクのマッピングを変更する必要があるかもしれません。たとえばブートローダ側の設定では `hda2` にルートディスクが存在しているように記述されているものの、KVM では `sda2` に現れるような場合、下記のようにブートローダのコマンドラインを変更する必要があります:

```
[...] root=/dev/hda2 resume=/dev/hda1 [...]
```

上記を下記のように変更します:

```
[...] root=/dev/sda2 resume=/dev/sda1 [...]
```

準仮想化型の `xvda` デバイスを使用している場合は、下記のようにします:

```
[...] root=/dev/vda2 resume=/dev/vda1 [...]
```

上記の変更を行わないと、VM ゲスト を KVM 環境で起動することができなくなります。

17.2.4.3 移行先での KVM ゲスト設定

上述の変更点を全て適用すると、KVM ゲストの設定は下記の例のようになります:

```
<domain type='kvm'>
  <name>SLES11SP3</name>
  <uuid>fa9ea4d7-8f95-30c0-bce9-9e58ffcabeb2</uuid>
  <memory>524288</memory>
  <currentMemory>524288</currentMemory>
  <vcpu cpuset='0-3'>1</vcpu>
  <os>
    <type arch=”x86_64”>hvm</type>
    <boot dev="hd"/>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type="raw"/>
      <source file='/var/lib/libvirt/images/SLES_11_SP2_JeOS.x86_64-0.0.2_para.raw'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <interface type='bridge'>
      <mac address='00:16:3e:2d:91:c3'/>
      <source bridge='br0'/>
    </interface>
    <input type='mouse' bus='usb'/>
    <graphics type='vnc' port='5900' autoport='yes' keymap='en-us'/>
    <memballoon model="virtio"/>
  </devices>
</domain>
```

変更後の設定は、ホームディレクトリ内に `SLES11SP3.xml` のようなファイル名で保存してください。取り込みを行うと、設定は `/etc/libvirt/qemu` ディレクトリ内にコピーされます。

17.2.5 VM ゲスト の移行

VM ゲスト の設定とゲスト OS 内での設定を変更したら、移行元の Xen ゲストをシャットダウンして、KVM ハイパーバイザで移行先を起動してください。

1. Xen ホスト内で動作するゲストをシャットダウンするには、コンソールから `root` で `shutdown -h now` を実行します。
2. 必要であれば、VM ゲスト に結びつけられているディスクファイルをコピーします。既定の設定では、Xen のディスクイメージは `/var/lib/xen/images` にありますので、これを `/var/lib/kvm/images` にコピーしてください。なお、KVM 側にゲストがまだ存在していない場合、`/var/lib/kvm/images` ディレクトリは存在しませんので、(`root` で) 作成してからコピーしてください。
3. 新しいドメインを作成して `libvirt` に登録します:

```
> sudo virsh define SLES11SP3.xml
 ドメイン SLES11SP3 が SLES11SP3.xml から定義されました
```

4. 作成したドメインが KVM の設定内に存在していることを確認します:

```
> virsh list -all
```

5. ドメインを作成したら、あとは起動するだけです:

```
> sudo virsh start SLES11SP3
 ドメイン SLES11SP3 が起動されました
```

17.3 さらなる情報

libvirt に関する詳細は、https://libvirt.org をお読みください。

`libvirt` の XML 書式に関する詳細については、https://libvirt.org/formatdomain.html をお読みください。

III 全ハイパーバイザ共通の機能

18 ディスクのキャッシュモード 211

19 VM ゲスト の時刻設定 215

20 libguestfs 217

21 QEMU ゲストエージェント 230

22 ソフトウエア TPM エミュレータ 233

23 VM ゲスト に対するクラッシュダンプの作成 236

18 ディスクのキャッシュモード

改訂履歴

2024-06-27

18.1 ディスクキャッシュとは

ディスクキャッシュは、ハードディスクとの間での読み込みや書き込みの速度を向上させるために使用するメモリです。物理的なハードディスクの場合は標準機能としてディスクキャッシュが存在しますが、仮想ディスクの場合は VM ホストサーバ のメモリを使用することになりますので、種類などを設定して細かい調整を行うことができるようになっています。

18.2 ディスクキャッシュの動作

通常、ディスクキャッシュは最もよく使用されるプログラムやデータを保持します。ユーザやプログラムがデータを要求すると、オペレーティングシステムは最初にディスクキャッシュ内を検索し、そこにデータがあれば、ディスクから読み込み直すことなくデータを渡すことができるようになっています。

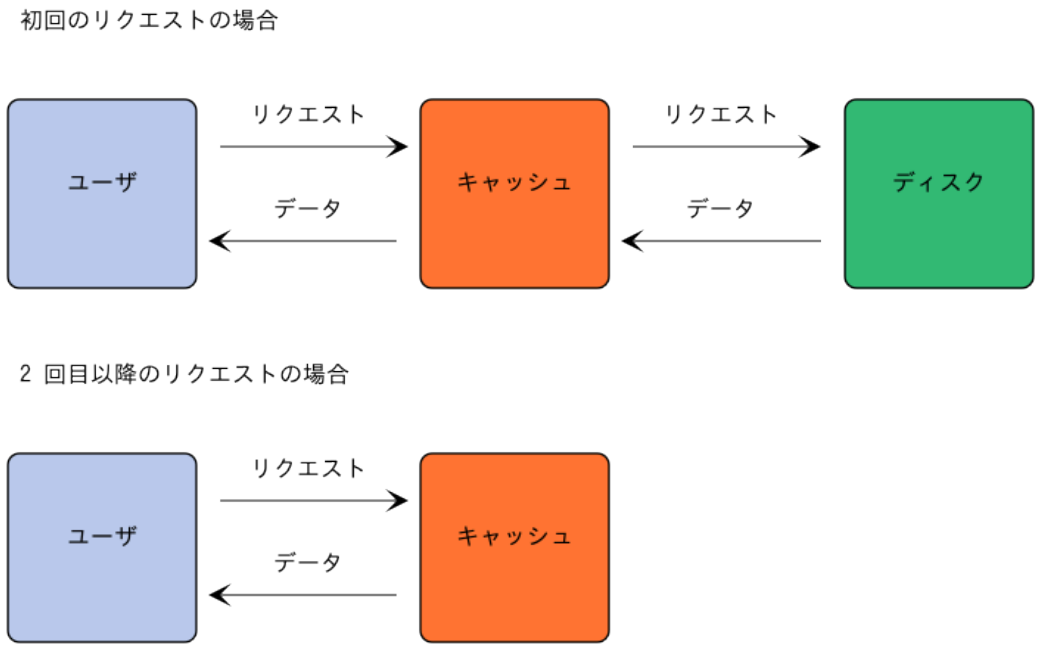


図 18.1: キャッシュの仕組み

18.3 ディスクキャッシュの利点

仮想ディスクデバイスのキャッシュを行うことで、ゲストマシンの性能を全体的に向上させることができます。キャッシュモードとディスクイメージ形式、そしてストレージサブシステムの組み合わせを最適化して、性能を向上させてください。

18.4 ディスクのキャッシュモード

キャッシュモードを指定しない場合、既定では `writeback` が使用されます。それぞれのゲストディスクに対しては、下記のいずれかのキャッシュモードを設定することができます:

writeback

`writeback` はホスト側のページキャッシュを使用するための設定です。ゲスト側からの書き込み要求は、ホスト側のキャッシュに配置された時点で完了とみなされます。あとはキャッシュ管理の仕組みによって、実際のストレージに書き込みを行います。ゲスト側の仮想ストレージアダプタに対しては ライトバック キャッシュが存在しているものとして通知するため、ゲスト側からはデータの一貫性を確保するために、必要に応じてフラッシュ (キャッシュの清掃) コマンドが送られるようになります。

writethrough

書き込み処理は、ストレージデバイスへのデータ書き込みをもって完了とみなします。ゲスト側の仮想ストレージアダプタに対しては ライトバック キャッシュが存在していないものとして通知されるため、ゲスト側からはフラッシュ (キャッシュの清掃) コマンドは送られません。

none

ホスト側のキャッシュを迂回し、ハイパーバイザとストレージデバイスとの間で読み込みおよび書き込みを直接行うようになります。実際のストレージデバイスでは、書き込みキュー内にのみデータを保存した時点で完了とみなすものも存在することから、ゲスト側の仮想ストレージアダプタに対しては ライトバック キャッシュが存在しているものとして通知します。このモードはホスト側のディスクに直接アクセスするのと同じ動作になります。

unsafe

writeback モードと似たような動作になりますが、ゲスト側からの全てのフラッシュ (キャッシュの清掃) コマンドは無視されます。このモードを使用すると性能が向上する代わりに、ホスト側で障害が発生した際にデータ損失が発生する可能性があります。このモードはゲストのインストール時に便利な仕組みであり、本番環境での使用には適しません。

directsync
: ゲスト側への書き込み完了はストレージデバイスへの書き込み完了まで待機させられ、ホスト側のキャッシュは迂回するようになります。writethrough モードと同様に、このモードはフラッシュ (キャッシュの清掃) コマンドを送信しないゲストに対して有用です。

18.5 キャッシュモードとデータの一貫性

writethrough, none, directsync
: これらのモードは、ゲスト側のオペレーティングシステムから必要に応じてフラッシュ (キャッシュの清掃) コマンドを受信するため、最も安全な選択肢として用意されています。安全ではないゲストや不安定なゲストをお使いの場合は、writethough または directsync をお使いください。

writeback
: このモードはゲスト側に対して書き込みキャッシュの存在を通知するため、ゲスト側からは必要に応じてフラッシュ (キャッシュの清掃) コマンドを送ってもらう必要があります。このモードは、ゲスト側から書き込み完了を受け取ったあと、ホスト側でディスクへの書き込みが完了するまでに時間的な遅延を生じさせる設定であることから、データ損失の可能性がありうることになります。

unsafe
: このモードは writeback 方式に似ていますが、ゲスト側からのフラッシュ (キャッシュの清掃) コマンドは全て無視されます。そのため、ホスト側の障害発生時、データ損失の危険性があることになります。

18.6 キャッシュモードとライブマイグレーションの関係

ストレージデータのキャッシュは、ライブマイグレーションに対応する際には障害となりうるものです。現時点では、ライブマイグレーションを行うには `raw` , `qcow2` のいずれかのイメージ形式を使用する必要があります。クラスタ型のファイルシステムを使用している場合は、全てのキャッシュモードでライブマイグレーションを行うことができます。それ以外の場合、読み書き可能な共有ストレージで使用できるキャッシュモードは `none` のみとなります。

`libvirt` の管理レイヤには、いくつかの要素をベースにした移行の互換性チェック機能が含まれています。ゲスト側のストレージがクラスタ型のファイルシステム内に存在している場合、それが読み込み専用か共有可能としてマークされていれば、移行が許可されるかどうかの判断ではキャッシュモードを考

慮しなくなります。それ以外の場合、`libvirt` ではキャッシュモードが `none` に設定されていない限り、移行を拒否するようになっています。ただし、移行時に「--unsafe」オプションを指定することで、この制限を無視することができます。`virsh` を利用して移行する場合は、下記のようにします:

```
> virsh migrate --live --unsafe
```



ヒント

AIO モード設定が `native` の場合、キャッシュモードを `none` に設定する必要があります。それ以外のキャッシュモードを使用している場合、AIO モードは自動的に既定値である `threads` に切り戻されます。

19 VM ゲスト の時刻設定

改訂履歴

2024-06-27

VM ゲスト で正しい時刻を維持することは、仮想化においてさらに難しい要素の 1 つになっています。時刻の維持はネットワークアプリケーションで特に重要であるほか、VM ゲスト のライブマイグレーションを実施する際の事前要件でもあります。



ヒント: VM ホストサーバ 側での時刻維持について

VM ホストサーバ 側でも同様に、NTP などを利用 (詳しくは『リファレンス』、第18章「NTP を利用した時刻同期」をお読みください) して、正しい時刻を維持するようにすることを強くお勧めします。

19.1 KVM: kvm_clock を使用する方法

KVM では、`kvm_clock` ドライバを介した準仮想化クロックに対応しています。`kvm_clock` を使用することを強くお勧めします。

下記のように実行すると、Linux の動作する VM ゲスト 側で `kvm_clock` ドライバを読み込んでいるかどうかを調べることができます:

```
> sudo dmesg | grep kvm-clock
[    0.000000] kvm-clock: cpu 0, msr 0:7d3a81, boot clock
[    0.000000] kvm-clock: cpu 0, msr 0:1206a81, primary cpu clock
[    0.012000] kvm-clock: cpu 1, msr 0:1306a81, secondary cpu clock
[    0.160082] Switching to clocksource kvm-clock
```

現在使用しているクロックソースを確認したい場合は、下記のコマンドを VM ゲスト 内で実行してください。`kvm-clock` と出力されるはずです:

```
> cat /sys/devices/system/clocksource/clocksource0/current_clocksource
```



重要: kvm-clock と NTP について

VM ゲスト 内で `kvm-clock` を使用している場合でも、NTP を併用することをお勧めします。VM ホストサーバ 側でも NTP の使用をお勧めします。

19.1.1 その他の時刻維持方式

準仮想化ドライバ `kvm-clock` は Windows* オペレーティングシステムには対応していません。Windows* をお使いの場合、時刻同期は `Windows Time サービスツール` を利用して行ってください。

19.2 Xen 仮想マシン時計設定

Xen 4 では、Xen ホストとゲストの間で時刻同期を行っていた `/proc/sys/xen/independent_wallclock` の設定が削除されました。また、それに代わって新しい `tsc_mode` というオプションが提供されるようになりました。これにより、Xen サーバとゲストとの間の時刻同期に際して、タイムスタンプカウンタ を使用する方法を指定できるようになっています。既定値は '0' で、大多数のハードウエアおよびソフトウエア環境でうまく動作するようになっています。

`tsc_mode` の詳細について、詳しくは `xen-tscmode` のマニュアルページをお読みください (`man 7 xen-tscmode`)。

20 libguestfs

改訂履歴

2025-01-22

仮想マシン にはディスクイメージと対応する定義ファイルが存在しています。通常のハイパーバイザを経由せずにゲスト側のコンポーネントにアクセスしたり、それらを変更したりすることもできますが、このような作業は本質的に危険なものであり、データの一貫性を破壊してしまうリスクをはらんでいます。libguestfs は C 言語で書かれたライブラリで、仮想マシン のディスクイメージに対して安全な方法で、アクセスしたり変更したりするための仕組みを提供します。このライブラリを使用することで、ハイパーバイザを使用することなく、かつ手作業による編集のリスクを伴うことなく、作業を行うことができます。

重要

libguestfs の使用は、AMD64/Intel 64 アーキテクチャのみで完全サポートされています。

20.1 VM ゲスト のディスク編集の概要

20.1.1 VM ゲスト のディスク編集によるリスク

ディスクイメージファイルと定義ファイルは、いずれも Linux 環境では単純なファイルとして存在しているものです。そのため、さまざまなツールを利用してアクセスしたり、編集や書き込みを行ったりすることができます。正しく使用している限りにおいては、これらのツールを利用することでゲスト側を管理す

ることができるようになります。しかしながら、これらのツールを正しく使用していても、リスク無しにディスクイメージを編集することはできません。ゲスト側のディスクイメージを編集する場合、下記のようなリスクが発生します:

- データの破壊：ホストマシンとクラスタ内の別ノードで同時にアクセスしてしまうと、仮想化による保護レイヤを迂回してしまうことになるため、データが失われたり破壊されたりしてしまいます。
- セキュリティ：ループバックデバイスを利用したディスクイメージのマウントは、root 権限が必要な作業です。ただし、イメージのマウントが成功してしまうと、他のユーザからディスクの内容にアクセスできてしまう場合があります。
- 管理上の作業ミス：仮想化レイヤの迂回には仮想化のコンポーネントやツールに対する高度な理解が必要となります。イメージを仮想化コンポーネントから切り離して編集を行い、編集完了後に仮想化コンポーネントに戻す作業は、時として作業ミスを引き起こしやすくなっています。

20.1.2 libguestfs の設計

libguestfs は C 言語で記述されたライブラリで、仮想マシン (VM ゲスト) のディスクイメージを安全に作成したり、アクセスや修正を行ったりする機能を提供するように設計されています。また libguestfs には、Perl (https://libguestfs.org/guestfs-perl.3.html) ↗ , Python (https://libguestfs.org/guestfs-python.3.html) ↗ , Ruby (https://libguestfs.org/guestfs-ruby.3.html) ↗ の言語バインディングも用意されています。libguestfs は root 権限を使用せずに VM ゲスト のディスクイメージにアクセスする機能を提供しているほか、不正なディスクイメージへのアクセスを防ぐための多階層の防御を提供しています。

libguestfs は VM ゲスト のディスクイメージとその内容にアクセスしたり、それらを修正したりするための多数のツールを提供しています。これらのツールには、ゲスト内のファイルを表示したり編集したりするツールのほか、スクリプトを利用して VM ゲスト の変更を行ったり、ディスクの使用率に関する統計情報の監視やゲストの作成、V2V もしくは P2V の移行を行う機能、バックアップの実施や VM ゲスト の複製、ディスクのフォーマットやサイズ変更などの機能が含まれています。



警告: 注意事項について

動作中の仮想マシンが使用しているディスクイメージに対しては、libguestfs ツールを使用してはなりません。動作中の状態のままディスクイメージにアクセスしてしまうと、VM ゲスト のディスクを破壊する結果になります。libguestfs ツールからアクセスしようとすると、対象の仮想マシンが動作中であればエラーを表示しようとしますが、完全な仕組みではないことに注意してください。

なお、ほとんどのコマンドには `--ro` (読み込み専用) オプションが用意されています。このオプションを指定すると、動作中の仮想マシンのディスクイメージにアクセスすることができます。ただし、仮想マシン側の動作によってファイルが正しく読み込めなくなってしまうことがありますが、ディスクが破壊されてしまうリスクはありません。

20.2 パッケージのインストール

libguestfs は 4 個のパッケージから構成されています:

- `libguestfs0` : メインの C 言語ライブラリです。
- `guestfs-data` : イメージを起動する際に使用するアプライアンスファイルが含まれています (`/usr/lib64/guestfs` ディレクトリ内に保存されています) 。
- `guestfs-tools` : 中枢となる guestfs ツールやマニュアルページ、そして `/etc/libguestfs-tools.conf` の設定ファイルが含まれています。
- `guestfs-winsupport` : guestfs ツールから Windows ゲストのファイルにアクセスするための機能が含まれています。このパッケージは、Windows ゲストを KVM に移行する場合など、Windows ゲストにアクセスする場合にのみ必要となります。

guestfs ツールをお使いのシステムにインストールするには、下記を実行します:

```
> sudo zypper in guestfs-tools
```

20.3 guestfs ツール

20.3.1 仮想マシンの修正

guestfs-tools パッケージ内にあるツールを使用することで、仮想マシンのディスクイメージにアクセスしたり、ディスクイメージを編集したりすることができます。この機能は、ディスクイメージの一貫性を損なうことがないようにするセーフガード付きのわかりやすいシェルインターフェイスを介して提供されていて、ここから guestfs API の全ての機能を利用することができます。また、マシンにインストールされているファイルと `/usr/lib64/guestfs` 内のファイルを使用することで、その場でアプライアンスを作成することもできます。

20.3.2 対応するファイルシステムとディスクイメージ

guestfs ツールは下記のファイルシステムに対応しています:

- Ext2, Ext3, Ext4
- Xfs
- Btrfs

複数のディスク形式にも対応しています:

- raw
- qcow2



警告: 未対応のファイルシステムについて

guestfs では Windows* のファイルシステム (VFAT, NTFS) のほか、BSD* や Apple* のファイルシステムにも対応していますし、その他のディスクイメージ形式 (VMDK, VHDX) にも対応しています。ですが、これらのファイルシステムとディスクイメージ形式は、いずれも openSUSE Leap ではサポート対象外となります。

20.3.3 `virt-rescue`

`virt-rescue` はレスキュー CD と同様の機能を提供する仕組みですが、仮想マシン向けに動作し、CD を使用せずに作業を行うことができます。`virt-rescue` はユーザに対してレスキューシェルを提供し、仮想マシンやディスクイメージ内に存在する問題を調査し、修正する機能を提供します。

```
> virt-rescue -a sles.qcow2
Welcome to virt-rescue, the libguestfs rescue shell.

Note: The contents of / are the rescue appliance.
You need to mount the guest's partitions under /sysroot
before you can examine them. A helper script for that exists:
mount-rootfs-and-chroot.sh /dev/sda1

><rescue>
[   67.194384] EXT4-fs (sda1): mounting ext3 file system
using the ext4 subsystem
[   67.199292] EXT4-fs (sda1): mounted filesystem with ordered data
mode. Opts: (null)
mount: /dev/sda1 mounted on /sysroot.
mount: /dev bound on /sysroot/dev.
```

```
mount: /dev/pts bound on /sysroot/dev/pts.
mount: /proc bound on /sysroot/proc.
mount: /sys bound on /sysroot/sys.
Directory: /root
Thu Jun  5 13:20:51 UTC 2014
(none):~ #
```

これで VM ゲスト をレスキューモードでアクセスすることができるようになります:

```
(none):~ # cat /etc/fstab
devpts   /dev/pts          devpts   mode=0620,gid=5 0 0
proc     /proc             proc     defaults        0 0
sysfs    /sys              sysfs    noauto          0 0
debugfs /sys/kernel/debug debugfs noauto          0 0
usbfs    /proc/bus/usb     usbfs    noauto          0 0
tmpfs    /run              tmpfs    noauto          0 0
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1 / ext3 defaults 1 1
```

20.3.4 `virt-resize`

`virt-resize` は仮想マシンのディスクのサイズを変更するためのツールで、サイズの拡張や縮小だけでなく、ディスク内のパーティションのサイズ変更や削除にも対応しています。

手順 20.1: ディスクのサイズ拡張

手順例: 仮想マシンのディスクサイズの拡張

1. まずは仮想マシンの電源を落としてシャットオフ状態にします。その後、仮想マシンのディスクイメージ内にあるパーティションのサイズを読み取ります:

   ```
   > virt-filesystems --long --parts --blkdevs -h -a sles.qcow2
   Name       Type       MBR  Size  Parent
   /dev/sda1  partition  83   16G   /dev/sda
   /dev/sda   device     -    16G   -
   ```

2. `virt-resize` はディスクイメージを直接書き換えることはできません。その代わり、サイズ変更後のディスクを用意して対応します。たとえば `truncate` コマンドを利用して、適切なサイズの仮想ディスクを作成します:

   ```
   > truncate -s 32G outdisk.img
   ```

3. あとは `virt-resize` コマンドを利用して、ディスクイメージのサイズを変更します。このとき、入力元と出力先のディスクイメージをそれぞれ指定する必要があることに注意してください:

   ```
   > virt-resize --expand /dev/sda1 sles.qcow2 outdisk.img
   ```

```
Examining sles.qcow2 ...
**********
Summary of changes:

/dev/sda1: This partition will be resized from 16,0G to 32,0G.  The
    filesystem ext3 on /dev/sda1 will be expanded using the 'resize2fs'
    method.

**********
Setting up initial partition table on outdisk.img ...
Copying /dev/sda1 ...
◐ 84%
#▒▒▒
```

20.3.5.2 virt-ls

`virt-ls` コマンドは、仮想マシンやディスクイメージ内にあるファイルの名前やサイズ、チェックサムや拡張属性などを出力します。複数のディレクトリを指定することもでき、この場合はそれぞれの出力が繋げられて出力されます。libvirt のゲスト側からディレクトリの一覧を表示したい場合は、`-d` オプションを指定してゲストの名前を入力してください。ディスクイメージから表示したい場合は、`-a` オプションをお使いください。

```
> virt-ls -h -lR -a sles.qcow2 /var/log/
d 0755        776 /var/log
- 0640          0 /var/log/NetworkManager
- 0644         23K /var/log/Xorg.0.log
- 0644         23K /var/log/Xorg.0.log.old
d 0700        482 /var/log/YaST2
- 0644        512 /var/log/YaST2/_dev_vda
- 0644         59 /var/log/YaST2/arch.info
- 0644        473 /var/log/YaST2/config_diff_2017_05_03.log
- 0644       5.1K /var/log/YaST2/curl_log
- 0644       1.5K /var/log/YaST2/disk_vda.info
- 0644       1.4K /var/log/YaST2/disk_vda.info-1
[...]
```

20.3.5.3 virt-cat

`virt-cat` はファイルの内容を出力するためのコマンドラインツールで、仮想マシンの名前やディスクイメージを指定して処理を行います。複数のファイル名を指定した場合は、それぞれが繋げられた状態で出力されます。また、ファイル名はフルパスでなければなりません (つまり、'/' で始まるルートディレクトリからのパスを指定する必要があります) 。

```
> virt-cat -a sles.qcow2 /etc/fstab
devpts /dev/pts devpts mode=0620,gid=5 0 0
proc   /proc    proc   defaults        0 0
```

20.3.5.4 virt-df

`virt-df` は仮想マシンのファイルシステム内での空き容量を表示するためのコマンドラインツールです。その他のツールとは異なり、仮想マシンに割り当てられたサイズを表示するだけでなく、ディスクイメージ内でどれだけの領域を使用しているのかを表示することができます。

```
> virt-df -a sles.qcow2
ファイルシステム                        1K-ブロック   使用       使用可能   使用%
sles.qcow2:/dev/sda1                    16381864     520564   15022492   4%
```

20.3.5.5 virt-edit

`virt-edit` は仮想マシン (またはディスクイメージ) 内に存在するファイルを編集するためのコマンドラインツールです。

20.3.5.6 virt-tar-in/out

`virt-tar-in` は非圧縮の TAR 書庫を仮想マシンのディスクイメージもしくは指定した名前の仮想マシンに展開します。`virt-tar-out` はその逆で、仮想マシンのディスクイメージや仮想マシンのディレクトリを TAR 書庫にまとめます。

```
> virt-tar-out -a sles.qcow2 /home homes.tar
```

20.3.5.7 virt-copy-in/out

`virt-copy-in` はローカルのディスク内にあるファイルやディレクトリを仮想マシンのディスクイメージもしくは指定した名前の仮想マシンにコピーします。`virt-copy-out` はその逆で、仮想マシンのディスクイメージもしくは指定した名前の仮想マシンから、ファイルやディレクトリをローカルのディスクにコピーします。

```
> virt-copy-in -a sles.qcow2 data.tar /tmp/
> virt-ls -a sles.qcow2 /tmp/
.ICE-unix
.X11-unix
data.tar
```

20.3.5.8 virt-log

`virt-log` は指定した名前の libvirt 仮想マシンもしくはディスクイメージ内にあるログファイルを表示します。`guestfs-winsupport` パッケージがインストールされていれば、Windows の仮想マシンディスクイメージから、イベントログの内容を表示することもできます。

```
> virt-log -a windows8.qcow2
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Events>
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event"><System><Provider
 Name="EventLog"></Provider>
<EventID Qualifiers="32768">6011</EventID>
<Level>4</Level>
```

```
<Task>0</Task>
<Keywords>0x0080000000000000</Keywords>
<TimeCreated SystemTime="2014-09-12 05:47:21"></TimeCreated>
<EventRecordID>1</EventRecordID>
<Channel>System</Channel>
<Computer>windows-uj49s6b</Computer>
<Security UserID=""></Security>
</System>
<EventData><Data><string>WINDOWS-UJ49S6B</string>
<string>WIN-KG190623QG4</string>
</Data>
<Binary></Binary>
</EventData>
</Event>

...
```

20.3.6 `guestfish`

`guestfish` は仮想マシンのファイルシステムを調査したり修正したりするためのシェルおよびコマンドラインツールです。libguestfs を使用し、guestfs API の全ての機能を提供します。

使用例:

```
> guestfish -a disk.img <<EOF
run
list-filesystems
EOF
```

```
guestfish

Welcome to guestfish, the guest filesystem shell for
editing virtual machine filesystems and disk images.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell

><fs> add sles.qcow2
><fs> run
><fs> list-filesystems
/dev/sda1: ext3
><fs> mount /dev/sda1 /
 cat /etc/fstab
devpts  /dev/pts          devpts  mode=0620,gid=5 0 0
proc    /proc             proc    defaults        0 0
sysfs   /sys              sysfs   noauto          0 0
debugfs /sys/kernel/debug debugfs noauto          0 0
```

```
usbfs   /proc/bus/usb     usbfs   noauto          0 0
tmpfs   /run              tmpfs   noauto          0 0
/dev/disk/by-id/ata-QEMU_HARDDISK_QM00001-part1 / ext3 defaults 1 1
```

20.3.7 物理イメージから KVM ゲストへの変換

libguestfs には、Xen の仮想マシンや物理マシンを KVM のゲストに変換する機能が用意されています。下記の章では、物理マシン (ベアメタルマシン) を KVM の仮想マシンに変換する特殊な使用例を説明しています。

ただし、物理マシンから KVM 仮想マシンへの変換は、openSUSE Leap ではサポート対象外となっており、技術プレビューとしての提供にとどまっています。

物理マシンの変換に際しては、まず物理マシンに関する情報収集を行い、それを変換サーバに送信する必要があります。この作業は、対象のマシン内で `virt-p2v` と KIWI NG を実行することで行うことができます。

手順 20.2: VIRT-P2V の使用

1. まずは必要なパッケージをインストールします:

   ```
   > sudo zypper in virt-p2v kiwi-desc-isoboot
   ```

   注記

   ここで説明している手順は、起動可能な DVD を作成するための ISO イメージの作成手順となります。この方法以外にも、PXE 起動イメージを作成する方法もあります。KIWI NG での PXE 起動イメージの構築方法について、詳しくは `man virt-p2v-make-kiwi` をお読みください。

2. KIWI NG の設定を作成します:

   ```
   > virt-p2v-make-kiwi -o /tmp/p2v.kiwi
   ```

   `-o` オプションを指定することで、KIWI NG の設定ファイルの出力先を指定しています。

3. 生成された設定内にある `config.xml` ファイルを必要に応じて修正します。たとえば `config.xml` ファイルを編集して、システムのキーボードレイアウトを調整するなどの処理があります。

4. `kiwi` を利用して ISO イメージを構築します:

   ```
   > kiwi --build /tmp/p2v.kiwi❶ \
   ```

```
    -d /tmp/build❷ \
    --ignore-repos \
    --add-repo http://(リポジトリのパス)❸ \
    --type iso
```

❶ 以前の手順で生成した KIWI NG 設定ディレクトリの場所を指定します。

❷ KIWI NG が ISO イメージを生成したり、途中のイメージを書き込んだりする際の出力先ディレクトリを指定します。

❸ `zypper lr -d` コマンドで出力することのできる、パッケージリポジトリの URL を指定します。
なお、リポジトリごとに `--add-repo` オプションを指定してください。

5. ISO イメージを DVD もしくは USB メモリに書き込みます。このメディアから起動を行うことで、変換対象のマシンを起動することができます。

6. システムを起動すると、変換サーバ の接続詳細を尋ねられます。このサーバは、`virt-v2v` パッケージがインストールされたマシンを意味します。
DHCP クライアントよりも複雑なネットワーク設定を行っている場合は、［ネットワークの設定］ボタンを押して、YaST のネットワーク設定ダイアログを開いてください。
［接続のテスト］ボタンを押して、ウイザードの次の手順に進みます。

7. 変換対象のディスクとネットワークインターフェイスを選択したあと、CPU の割当数やメモリ、仮想マシン名などの仮想マシン向けのデータを設定します。



注記

何も指定しない場合、作成されたディスクイメージ形式は既定で raw になります。これは［出力形式］の項目内で形式を指定することで、変更を行うことができます。

仮想マシンの生成方法には 2 種類の仕組みがあります。それぞれ ローカル および libvirt 出力と呼ばれます。前者の場合、仮想マシンのディスクイメージと設定の出力先を［出力ストレージ］の項目に指定します。これらは `virsh` を利用することで、libvirt の管理する仮想マシンにすることができます。後者の場合は［出力ストレージ］内に定義されたプール内にディスクイメージを配置して、新しく libvirt の管理する仮想マシンを作成します。
［変換開始］を押して開始してください。

20.4 トラブルシューティング

20.4.1 btrfs 関連の問題

btrfs をルートパーティションとしているイメージ (openSUSE Leap の既定値です) に対して guestfs ツールを使用すると、下記のようなエラーメッセージが表示されることがあります:

```
> virt-ls -a /path/to/sles12sp2.qcow2 /
virt-ls: multi-boot operating systems are not supported

If using guestfish '-i' option, remove this option and instead
use the commands 'run' followed by 'list-filesystems'.
You can then mount file systems you want by hand using the
'mount' or 'mount-ro' command.

If using guestmount '-i', remove this option and choose the
filesystem(s) you want to see by manually adding '-m' option(s).
Use 'virt-filesystems' to see what file systems are available.

If using other virt tools, multi-boot operating systems won't work
with these tools.  Use the guestfish equivalent commands
(see the virt tool manual page).
```

これは一般に、ゲスト内にスナップショットが存在することによって発生します。この場合、どのスナップショットから起動すべきなのかが判断できないことを表しています。特定のスナップショットを指定して起動する場合は、`-m` オプションを指定してください:

```
> virt-ls -m /dev/sda2:/:subvol=@/.snapshots/2/snapshot -a /path/to/sles12sp2.qcow2 /
```

20.4.2 環境

libguestfs アプライアンス内で問題を分析する場合、環境変数 LIBGUESTFS_DEBUG=1 を指定すると、デバッグメッセージを出力することができます。出力されるコマンドや API コールを guestfish でのコマンドに似た形式で出力したい場合は、環境変数 LIBGUESTFS_TRACE=1 を指定してください。

20.4.3 `libguestfs-test-tool`

`libguestfs-test-tool` ツールは、libguestfs の基本的な機能が動作しているかどうかを確認するためのテストプログラムです。大量の分析用メッセージと guestfs 環境詳細が出力され、テストイメージを作成して開始しようとします。テストが問題なく完了すると、下記のようなメッセージが末尾付近に現れるはずです:

```
===== TEST FINISHED OK =====
```

20.5 さらなる情報

- libguestfs.org (https://libguestfs.org)
- libguestfs FAQ (英語) (https://libguestfs.org/guestfs-faq.1.html)

21 QEMU ゲストエージェント

改訂履歴

2024-06-27

QEMU ゲストエージェント (GA) は VM ゲスト 内で動作するもので、VM ホストサーバ から `libvirt` を介して、ゲスト側でコマンドを実行するための仕組みです。この仕組みでは、ゲスト側のファイルシステムに関する情報やファイルシステムの一時休止／再開、ゲストの一時停止や再起動などを行うことができます。

QEMU GA は `qemu-guest-agent` パッケージ内に含まれています。このパッケージは、KVM の仮想マシン環境を設定していれば、既定でインストールおよび設定、有効化までが行われます。

QEMU GA は Xen 仮想マシンでもインストールされますが、既定では有効化されません。また、Xen 仮想マシンでも使用することができますが、KVM 仮想マシンのような libvirt との統合機能は提供されていません。QEMU GA を Xen で使用する場合は、VM ゲスト の設定でチャンネルデバイスを追加しなければなりません。チャンネルデバイスは VM ホストサーバ 側の Unix ドメインソケットで、QEMU GA との通信に使用するチャンネルです。

```
<channel type='unix'>
  <source mode='bind' path='/example/path'/>
  <target type='xen' name='org.qemu.guest_agent.0'/>
</channel>
```

21.1 QEMU GA コマンドの実行

QEMU GA には、`libvirt` では直接提供していない多数の組み込みコマンドが提供されています。完全な一覧については 21.4項「さらなる情報」をお読みください。また QEMU GA のコマンドは、`libvirt` の汎用コマンド `qemu-agent-command` を利用して実行します:

```
virsh qemu-agent-command ドメイン名 '{"execute":"QEMU_GA_のコマンド"}'
```

たとえば下記のようになります:

```
> sudo virsh qemu-agent-command sle15sp2 '{"execute":"guest-info"}' --pretty
{
"return": {
  "version": "4.2.0",
  "supported_commands": [
    {
      "enabled": true,
      "name": "guest-get-osinfo",
      "success-response": true
    },
```

```
[...]
```

21.2 QEMU GA を必要とする `virsh` のコマンド

`virsh` のコマンド内にも、実行するにあたって QEMU GA の機能を必要とするものがあります。下記にそれらのうちのいくつかを示します:

`virsh guestinfo`
: ゲスト側の観点で、ゲストに関する情報を表示します。

`virsh guestvcpus`
: ゲスト側の観点で、仮想 CPU の状態を問い合わせたり、設定を変更したりすることができます。

`virsh set-user-password`
: ゲスト内のユーザアカウントに対して、パスワードの設定を行います。

`virsh domfsinfo`
: 動作中のドメイン内で、マウントされているファイルシステムの一覧を表示します。

`virsh dompmsuspend`
: 動作中のゲストを一時停止します。

21.3 `libvirt` コマンドの拡張

ゲスト内で QEMU GA が有効化されていれば、エージェント モードで動作させることで、様々な機能を提供する多数の `virsh` サブコマンドを実行することができるようになります。下記の一覧には、それらのうちのいくつかの例を示します。完全な一覧を読みたい場合は、`virsh` のマニュアルページにて `agent` で検索を行ってご確認ください。

`virsh shutdown --mode agent` および `virsh reboot --mode agent`
: このサブコマンドを実行すると、次回以降のゲスト実行時に障害を発生させたりすることのないよう、適切なシャットダウンもしくは再起動を行うことができます。これは ACPI でのシャットダウンや再起動と似ています。

`virsh domfsfreeze` および `virsh domfsthaw`
: ゲストに対してファイルシステムの一時休止を指示します。これにより、キャッシュ内に残っている全ての I/O 操作を完了させ、ボリュームの一貫性が保たれた状態になります。そのため、再マウント時にファイルシステムのチェックが不要となります。

`virsh setvcpus --guest`
: ゲストに割り当てている CPU の数を変更します。

`virsh domifaddr --source agent`
: QEMU GA に対して、ゲスト側の IP アドレスを問い合わせます。

`virsh vcpucount --guest`
: ゲスト側の観点で、仮想 CPU の数に関する情報を表示します。

21.4 さらなる情報

- QEMU GA で提供している全てのコマンドの一覧については、https://www.qemu.org/docs/master/interop/qemu-ga-ref.html をお読みください。
- `virsh` のマニュアルページ (`man 1 virsh`) には、QEMU GA インターフェイスを利用することのできるコマンドの説明が示されています。

22 ソフトウエア TPM エミュレータ

改訂履歴

2024-06-27

22.1 概要

Trusted Platform Module (TPM) は暗号処理を行うプロセッサで、暗号鍵を利用してハードウエアの機密を保持するための仕組みです。TPM は開発者向けにセキュリティ機能を提供しますので、TPM を擬似的に再現するエミュレータが存在すると、より安全性の高い仮想環境を構築できるようになります。また、ハードウエア TPM デバイスとは異なり、ゲスト側からのアクセス数に制限がありません。このほか、TPM バージョン 1.2 と 2.0 を簡単に切り替えて使用することもできます。QEMU では `swtpm` パッケージで提供されるソフトウエア TPM エミュレータに対応しています。

22.2 事前要件

ソフトウエア TPM エミュレータをインストールして使用するには、まず `libvirt` 仮想化環境をインストールする必要があります。6.2項「仮想化コンポーネントのインストール」を参照して、提供されている仮想化ソリューションの中からいずれかをインストールしてください。

22.3 インストール

ソフトウエア TPM エミュレータを使用するには、`swtpm` パッケージをインストールします:

```
> sudo zypper install swtpm
```

22.4 QEMU での `swtpm` の使用

`swtpm` は 3 種類のインターフェイスを提供しています。それぞれ `socket` , `chardev` , `cuse` と呼ばれます。下記の手順では socket インターフェイスを使用するものとします。

1. VM のディレクトリ内に `mytpm0` というディレクトリを作成して、ここに TPM の状態を保存します。たとえば VM のディレクトリが `/var/lib/libvirt/qemu/sle15sp3` であれば、下記のようになります:

```
> sudo mkdir /var/lib/libvirt/qemu/sle15sp3/mytpm0
```

2. あとは `swtpm` を起動します。起動を行うと QEMU (例: `/var/lib/libvirt/qemu/sle15sp3`) が使用するソケットファイルが作成されます:

```
> sudo swtpm socket
 --tpmstate dir=/var/lib/libvirt/qemu/sle15sp3/mytpm0 \
 --ctrl type=unixio,path=/var/lib/libvirt/qemu/sle15sp3/mytpm0/swtpm-sock \
 --log level=20
```



ヒント: TPM バージョン 2.0 について

既定では `swtpm` は TPM バージョン 1.2 エミュレータを起動し、その状態を `tpm-00.permall` ディレクトリ内に保存します。TPM 2.0 インスタンスを作成したい場合は、下記のように入力して実行します:

```
> sudo swtpm socket
 --tpm2
 --tpmstate dir=/var/lib/libvirt/qemu/sle15sp3/mytpm0 \
 --ctrl type=unixio,path=/var/lib/libvirt/qemu/sle15sp3/mytpm0/swtpm-sock \
 --log level=20
```

TPM 2.0 の場合、状態は `tpm2-00.permall` ディレクトリ内に保存されます。

3. あとは `qemu-system-アーキテクチャ` コマンドに対して、下記のようなコマンドラインパラメータを追加します:

```
> qemu-system-x86_64 \
[...]
-chardev socket,id=chrtpm,path=/var/lib/libvirt/qemu/sle15sp3/mytpm0/swtpm-sock \
-tpmdev emulator,id=tpm0,chardev=chrtpm \
-device tpm-tis,tpmdev=tpm0
```

4. 最後にゲスト側で、TPM デバイスが利用可能になっているかどうかを確認します。具体的には下記のコマンドを入力して実行します:

```
> tpm_version
TPM 1.2 Version Info:
Chip Version:        1.2.18.158
Spec Level:          2
Errata Revision:     3
```

```
TPM Vendor ID:        IBM
TPM Version:          01010000
Manufacturer Info:    49424d00
```

22.5 libvirt での swtpm の使用

`libvirt` で swtpm を使用する場合は、ゲストの XML 設定内に TPM デバイスの情報を追加します:

```
<devices>
 <tpm model='tpm-tis'>
  <backend type='emulator' version='2.0'/>
 </tpm>
</devices>
```

`libvirt` の場合、swtpm はゲストの起動と共に自動的に開始されます。事前に起動しておく必要はありません。`permall` 状態ファイルは `/var/lib/libvirt/swtpm/VM_UUID` ディレクトリ内に保存されます。

22.6 OVMF ファームウエアでの TPM の完全性計測

ゲスト側で Open Virtual Machine Firmware (OVMF) を使用している場合、TPM の完全性計測が行われます。イベントログを読む場合は、`/sys/kernel/security/tpm0/binary_bios_measurements` ファイルを参照してください。

22.7 各種情報

- Wikipedia の https://ja.wikipedia.org/wiki/Trusted_Platform_Module には、TPM に関する大まかな説明が書かれています。
- openSUSE Leap 固有の仮想環境の設定については、第6章「仮想化コンポーネントのインストール」に説明があります。
- `swtpm` の使用方法に関する詳細は、マニュアルページ (`man 8 swtpm`) をお読みください。
- TPM の `libvirt` における仕様については、https://libvirt.org/formatdomain.html#elementsTpm (英語) に説明が書かれています。
- OVMF での UEFI ファームウエアの有効化に関する説明は、9.3.1項「高度な UEFI 設定」をお読みください。

23 VM ゲスト に対するクラッシュダンプの作成

改訂履歴

2024-06-27

23.1 概要

仮想マシンがクラッシュした場合、デバッグや分析などの目的で、メモリに対するコアダンプを採取しておくと便利です。物理マシンの場合は Kexec と Kdump がクラッシュダンプの採取を実施しますが、仮想マシンの場合は、完全仮想化マシンなのか準仮想化マシンなのかによって、異なる方式を使用します。

23.2 完全仮想化マシンに対するクラッシュダンプの作成

完全仮想化マシンに対してクラッシュダンプを採取する場合は、物理マシンと全く同じ方式 (Kexec および Kdump) を使用します。

23.3 準仮想化マシンに対するクラッシュダンプの作成

完全仮想化マシンとは異なり、準仮想化マシンでは Kexec/Kdump は動作しません。その代わりとして、ホストツールスタック側でクラッシュダンプを採取します。Xen domU で `xl` ツールスタックを使用している場合は `xl dump-core` コマンドを、`libvirt` ベースの VM ゲスト の場合は `virsh dump` コマンドを実行することで、ダンプを採取することができます。

それ以外にも、VM ゲスト の設定内で `on_crash` の設定をすることで、クラッシュダンプを自動的に採取させることもできます。この設定では、ホストツールスタック側に対して、クラッシュ発生時に行うべき処理を指定することもできます。なお、`xl` と `libvirt` のいずれも、既定値は `destroy` (仮想マシンを終了する) です。クラッシュダンプの自動採取を指定したい場合は、`coredump-destroy` (コアダンプを採取して停止する) もしくは `coredump-restart` (コアダンプを採取して再起動する) のいずれかを指定してください。

23.4 追加の情報

- 完全仮想化マシンと準仮想化マシンの違いについては、1.3項「仮想化モード」をお読みください。
- Kexec/Kdump の仕組みに関する詳細は、『システム分析／チューニングガイド』、第18章「Kexec と Kdump」をお読みください。
- `xl` の設定書式に関する詳細は、xl.cfg のマニュアルページ (`man 5 xl.cfg`) をお読みください。
- `libvirt` の XML 設定に関する詳細は、https://libvirt.org/formatdomain.html#events-configuration をお読みください。

IV Xen を利用した仮想マシンの管理

24 仮想マシンホストの設定 239

25 仮想ネットワーク 251

26 仮想環境の管理 260

27 Xen 内でのブロックデバイス 267

28 仮想化: オプション設定 271

29 管理作業 281

30 XenStore: ドメイン間で共有される設定データベース 290

31 Xen の高可用性仮想化ホストとしての使用 296

32 Xen: 準仮想化 (PV) ゲストから完全仮想化 (FV/HVM) ゲストへの変換 299

24 仮想マシンホストの設定

改訂履歴

2024-06-27

本章では、openSUSE Leap 15.7 を仮想マシンのホストとして設定し、それを使用するまでの手順を説明しています。

通常、Dom0 に対するハードウエア要件は特に存在せず、openSUSE Leap オペレーティングシステムの要件と同じになります。ただし、構築予定の VM ゲスト システムのリソース要件を全て満たすため、追加の CPU やディスク、メモリやネットワークリソースなどを用意しておくことをお勧めします。



ヒント: リソースについて

物理マシンと同様に、VM ゲスト システムでも、より高速なプロセッサとより大容量のメモリを割り当てておくことで、性能を向上させることができます。

仮想マシンのホスト側となるには、いくつかのソフトウエアパッケージのほか、それらが必要とする追加のソフトウエアパッケージをインストールする必要があります。必要なパッケージをインストールするには、YaST の［ソフトウエア管理］を起動し、［表示］›［パターン］を選択したあと、［Xen 仮想マシンホストサーバ］を選択してインストールを行ってください。この方法のほか、YaST の［仮想化］›［ハイパーバイザとツールのインストール］からでもインストールを行うことができます。

Xen 関係のソフトウエアをインストールしたあとはコンピュータを再起動し、ブートローダの画面で Xen カーネルのオプションを選択してください。

更新は通常の更新チャンネルを介して提供されます。最新の更新をインストールするには、インストール後に YaST 内の［オンライン更新］を実行してください。

24.1 注意事項

ホスト側に openSUSE Leap オペレーティングシステムをインストールして設定するには、下記の事項に注意してください:

- ホスト側では Xen ホストを動作させる必要があります。YaST を起動して［システム］›［ブートローダ］を選択して、Xen の起動項目を既定値に設定してください。

    - YaST を起動して［システム］›［ブートローダ］を選択します。
    - 既定の起動項目を［Xen］ラベル付きのものに設定します。
    - ［Ok］ボタンを押します。
- 性能を最大限に発揮するためには、仮想マシンホストには仮想化に必要なアプリケーションのみをインストールし、動作させておくことをお勧めします。
- Xen ホストに接続されているウオッチドッグデバイスを使用したいとお考えの場合、同時に複数台の VM ゲスト から使用することは避けてください。また、一般的なソフトウエア実装ではなく、実際のハードウエア統合のドライバをお使いください。



注記: ハードウエアの監視について

Dom0 カーネルは仮想化環境内で動作する仕組みであるため、`irqbalance` や `lscpu` などのコマンドが、実際のハードウエア特性を表さないかもしれません。



重要: Xen では Trusted Boot がサポートされない件について

Xen では Trusted Boot (Tboot) はサポート対象外となります。Xen を正しく起動できるようにするため、GRUB 2 の設定ダイアログでは［Trusted Boot サポート］の選択を外していることを確認してください。

24.2 Dom0 のメモリ管理

以前のバージョンの openSUSE Leap では、Dom0 に対して全ての物理メモリを割り当て、自動バルーン設定が有効化されるよう、Xen ホストのメモリ設定が行われていました。メモリはドメイン側で要求があるたびに Dom0 側から自動的に割り当てられる (バルーン) ようになっていました。この設定ではエラーを引き起こしやすくなってしまうことから、現在は無効化しておくことを強くお勧めしています。openSUSE Leap 15.1 およびそれ以降のバージョンでは、既定で自動バルーンが無効化されるほか、Dom0 に対してはホストの物理メモリの 10% + 1 GB 程度が割り当てられるようになっています。たとえば物理メモリが 32 GB のシステムで使用した場合、Dom0 には 4.2 GB 程度が割り当てられます。

なお、`/etc/default/grub` 内で `dom0_mem` という Xen コマンドラインオプションを使用することもできます。この方式も現在の推奨となっています。従来の動作に戻したい場合は、`dom0_mem` の設定を物理メモリサイズと同じ値に設定し、`/etc/xen/xl.conf` 内で `autoballoon` を有効化してください。

警告: Dom0 のメモリが不足する場合について

Dom0 向けに予約するメモリ量は、Dom0 がそれぞれの VM ゲスト に対してバックエンドのネットワークとディスク I/O サービスを提供することから、ホスト内で動作させる VM ゲスト の数に従って決まります。ただし、Dom0 のメモリ割り当てを計算する場合は、それ以外の負荷に対しても考慮を行うものとし、他の仮想マシンと同様にサイズを判断してください。

24.2.1 Dom0 のメモリ割り当ての設定

1. まずは Dom0 に対して必要なメモリ割当量を判断します。
2. Dom0 で `xl info` と入力して実行し、マシンで利用可能なメモリ量を表示します。Dom0 に対する現在のメモリ割り当ては、`xl list` で表示することができます。
3. `/etc/default/grub` ファイルを編集して、`GRUB_CMDLINE_XEN` オプション内に `dom0_mem=メモリ量` の形式で指定を行います。このとき、`メモリ量` には Dom0 に割り当てる最大のメモリ量を指定します。なお、末尾に `K` (キロバイト), `M` (メガバイト), `G` (ギガバイト) の単位接尾辞を指定することもできます。たとえば下記のようになります:

   ```
   GRUB_CMDLINE_XEN="dom0_mem=2G"
   ```

4. あとはコンピュータを再起動して、設定を反映させます。

ヒント

Xen 関連の起動設定オプションについて、詳しくは『リファレンス』、第12章「ブートローダ GRUB 2」、12.2.2項「`/etc/default/grub` ファイル」をお読みください。

警告: Xen の Dom0 メモリについて

XL ツールスタックと GRUB 2 内での Xen ハイパーバイザ向けの `dom0_mem=` オプションを併用している場合、`etc/xen/xl.conf` 内で autoballoon を無効化する必要があります。無効化しないと、VM の起動時に Dom0 のバルーン縮小を行うことができず、失敗することになってしまいます。そのため Xen 向けに `dom0_mem=` を指定した場合は、`xl.conf` 内で必ず autoballoon=0 を設定してください。詳しくは Xen dom0 memory (https://wiki.xen.org/wiki/Xen_Best_Practices#Xen_dom0_dedicated_memory_and_preventing_dom0_memory_ballooning) (英語) をお読みください。

24.3 完全仮想化環境でのネットワークカード

完全仮想化環境のゲストでは、既定で Realtek ネットワークカードの擬似版を使用します。しかしながら、Dom0 と VM ゲスト の間での通信が必要となる場合は、個別のネットワークドライバを使用することもできます。オペレーティングシステムによっては両方のインターフェイスを提供しておく必要があることから、既定では両方のインターフェイスを VM ゲスト 側に提供します。

openSUSE Leap を使用している場合、既定では VM ゲスト に対して、準仮想化ネットワークカードのみを提供します。この場合、下記のネットワークオプションを選ぶことができます:

擬似版
: 擬似版 Realtek カードのような擬似ネットワークインターフェイスを使用するには、ドメイン xl 設定の `vif` デバイスセクション内に、`type=ioemu` を指定します。たとえば下記のようになります:

```
vif = [ 'type=ioemu,mac=00:16:3e:5f:48:e4,bridge=br0' ]
```

`xl.conf` 内での xl 設定について、詳しくは `man 5 xl.conf` で表示されるマニュアルページをお読みください。

準仮想化版
: `type=vif` を指定して model や種類を指定しないと、準仮想化版のネットワークインターフェイスを使用します:

```
vif = [ 'type=vif,mac=00:16:3e:5f:48:e4,bridge=br0,backen=0' ]
```

擬似版と準仮想化版の両方
: 両方のオプションを必要とする場合は、単純に type と model の両方を指定します。xl の設定は下記のようになります:

```
vif = [ 'type=ioemu,mac=00:16:3e:5f:48:e4,model=rtl8139,bridge=br0' ]
```

この場合、VM ゲスト 内ではいずれかのネットワークインターフェイスを無効化する必要があります。

24.4 仮想マシンホストの開始

仮想化ソフトウエアを正しくインストールすることができたら、あとは GRUB 2 ブートローダで［Xen］オプション付きの項目を選択して Xen を読み込みます。仮想マシンのホスト側では必ず選択するようにしてください。

警告

なお、Xen システムを起動する際、dom0 の /var/log/messages もしくは `systemd` のジャーナルに、下記のようなエラーメッセージが記録される場合があります:

```
isst_if_mbox_pci: probe of 0000:ff:1e.1 failed with error -5
isst_if_pci: probe of 0000:fe:00.1 failed with error -5
```

上記は ISST ドライバが仮想マシンに対する電源制御や周波数制御の機能を提供していない旨を表しているだけですので、無視してかまいません。

注記: Xen と Kdump について

Xen ではハイパーバイザがメモリリソースを管理します。Dom0 内でリカバリカーネル用のシステムメモリ予約を行いたい場合、ハイパーバイザ側で予約を行う必要があります。そのため、`/etc/default/grub` ファイル内で指定する `GRUB_CMDLINE_XEN_DEFAULT` 内に `crashkernel=サイズ` パラメータを指定する必要があります。ファイルを保存したら、あとは下記のコマンドを実行します:

```
> sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

crashkernel パラメータに関する詳細は、**『システム分析／チューニングガイド』、第18章「Kexec と Kdump」、18.4項「crashkernel 割り当てサイズの計算」**をお読みください。

GRUB 2 のメニュー内に［Xen］オプションが存在しない場合、まずは前述のインストール手順を確認し、GRUB 2 のブートローダ設定が更新されていることを確認してください。Xen のパターンを選択せずにインストールを行っている場合は、YaST の［ソフトウエア管理］で［パターン］を選択し、［Xen 仮想マシンホストサーバ］を選択してインストールしてください。

ハイパーバイザを起動したあとは、Dom0 仮想マシンが起動し、グラフィカルなデスクトップ環境を表示します。グラフィカルなデスクトップ環境をインストールしていない場合は、コマンドライン環境が表示されます。

ヒント: グラフィックで発生しうる問題について

場合によってはグラフィックシステムが正しく動作しなくなってしまうことがあります。この場合、起動パラメータに `vga=ask` を追加してください。また、解像度を固定で指定したい場合は、`vga=mode-0x???` を指定してください。なお、`???` の値は `0x100` に VESA モー

ド番号を加えることで計算することができます。VESA モード番号については、https://en.wikipedia.org/wiki/VESA_BIOS_Extensions (英語のみ) に書かれています。たとえば `vga=mode-0x361` のように設定します。

仮想化ゲストをインストールする前に、まずはシステムの時刻が正しく設定されていることを確認してください。これを行うには、制御ドメイン側で NTP (Network Time Protocol) を設定します:

1. YaST を起動して［ネットワークサービス］›［NTP 設定］を選択します。
2. NTP デーモンを起動時に自動的に開始するように設定したあと、既知の NTP サーバを指定して［OK］を押します。

注記: 仮想化ゲストでの時刻サービスについて

ハードウエア側に搭載されている時計は非常に精度の低いものです。全ての新しいオペレーティングシステムでは、設定された時刻情報源から時刻を取得し、ハードウエアの時刻をシステムの時刻にあわせようとします。全ての VM ゲスト システムに対して正しい時刻を設定するには、それぞれのゲスト側でも NTP の設定を行うか、もしくはゲスト側に対してホストが提供する時刻を取得するよう設定してください。openSUSE Leap における **時刻設定** について、詳しくは 19.2項「Xen 仮想マシン時計設定」をお読みください。

仮想マシンの管理に関する詳細は、第26章「仮想環境の管理」をお読みください。

24.5 PCI パススルー

VM ゲスト システムを最大限に活用するには、特定の PCI デバイスを特定の VM ゲスト に割り当てる必要がある場合があります。完全仮想化ゲストでこの機能を使用する場合、この機能に対応するチップセットが必要となるほか、BIOS 側でも有効化しておく必要があります。

この機能は AMD* および Intel* の両方で使用することができます。AMD の場合、この機能は IOMMU と呼ばれ、Intel の場合は VT-d と呼ばれます。なお、Intel-VT 技術の場合、完全仮想化ゲストで使用するには不十分な機能しか提供されていません。お使いのコンピュータがこの機能に対応しているかどうかを知るには、システムの販売元や提供元にお尋ねください。

制限事項

- グラフィックドライバによっては、DMA へのアクセスを高度に最適化しているものがあり、サポート対象外となっているものがあります。そのため、グラフィックカードの使用については難しいものとお考えください。

- PCIe ブリッジ内にある PCI デバイスにアクセスする場合、全ての PCI デバイスを単一のホストに割り当てなければなりません。この制限は、PCIe デバイスには当てはまりません。
- 専用の PCI デバイスが割り当てられたゲストについては、異なるホストへの移行を行うことができません。

PCI パススルー の設定には 2 つのものがあります。1 つはハイパーバイザ側の設定で、再割り当てを行うことができるようにするための起動時の設定、もう 1 つは VM ゲスト 側での PCI デバイスの割り当て設定です。

24.5.1 PCI パススルー に対応するためのハイパーバイザ側の設定

1. まずは VM ゲスト に再割り当てするデバイスを選択します。`lspci -k` コマンドを実行してデバイスの一覧を表示し、デバイス番号のほか、デバイスに元々割り当てられていたモジュール名を記録しておきます:

   ```
   06:01.0 Ethernet controller: Intel Corporation Ethernet Connection I217-LM (rev 05)
           Subsystem: Dell Device 0617
           Kernel driver in use: e1000e
           Kernel modules: e1000e
   ```

   上記の例では、デバイス番号が `06:01.0` 、モジュール名が `e1000e` になります。

2. `xen_pciback` がデバイスを制御するための最初のモジュールとなるよう、モジュールの依存関係を指定します。`/etc/modprobe.d/50-e1000e.conf` というファイルを作成して、下記の内容を記述します:

   ```
   install e1000e /sbin/modprobe xen_pciback ; /sbin/modprobe \
    --first-time --ignore-install e1000e
   ```

3. `xen_pciback` モジュールに対して、`hide` オプションを使用してデバイスを制御するように指定します。`/etc/modprobe.d/50-xen-pciback.conf` というファイルを編集するか新規作成して、下記の内容を記述します:

   ```
   options xen_pciback hide=(06:01.0)
   ```

4. あとはシステムを再起動します。

5. 下記のコマンドを実行して、割り当て可能なデバイスとして対象のデバイスが表示されていることを確認します:

   ```
   xl pci-assignable-list
   ```

24.5.1.1 xl を利用した動的な割り当て

ホスト側のシステムの再起動を避けるには、PCI パススルー を使用する際に xl による動的な割り当てを使用することができます。

dom0 に対して pciback モジュールを読み込んでおきます:

```
> sudo modprobe pciback
```

あとは `xl pci-assignable-add` コマンドを実行して、デバイスを割り当て可能な状態にします。たとえばゲストに対して 06:01.0 のデバイスを割り当てられるようにするには、下記のコマンドを実行します:

```
> sudo xl pci-assignable-add 06:01.0
```

24.5.2 VM ゲスト システムに対する PCI デバイスの割り当て

VM ゲスト に対して PCI デバイスを専用に割り当てるには、いくつかの方法があります:

インストール時のデバイス追加

インストール時に設定ファイル内に `pci` 行を追加します:

```
pci=['06:01.0']
```

VM ゲスト システムへの PCI デバイスの活性接続 (ホットプラグ)

`xl` コマンドを使用することで、PCI デバイスの追加や削除をその場で行うことができます。たとえばデバイス番号が `06:01.0` で、仮想マシン名が `sles12` であれば、下記のように入力して実行します:

```
xl pci-attach sles12 06:01.0
```

Xend に対する PCI デバイスの追加

ゲストに対して恒久的にデバイスを追加するには、下記の内容をゲスト側の設定ファイルに追加します:

```
pci = [ '06:01.0,power_mgmt=1,permissive=1' ]
```

PCI デバイスを VM ゲスト 側に割り当てたあとは、ゲスト側で設定を行うほか、デバイスドライバについてもインストールを行ってください。

24.5.3 VGA パススルー

Xen 4.0 およびそれ以降のバージョンでは、完全仮想化環境の VM ゲスト に対して、VGA グラフィックアダプタのパススルーに対応するようになりました。これにより、ゲスト側で高性能な完全 3D 環境やビデオアクセラレーションを使用できるようになっています。

制限事項

- VGA パススルー の機能は PCI パススルー と同様の仕組みで動作するものであり、こちらについてもメインボード側のチップセットと BIOS の両方で IOMMU (もしくは Intel VT-d) に対応する必要があります。
- プライマリのグラフィックアダプタ (コンピュータの電源を入れた際に使用していたもの) のみを VGA パススルー として使用することができます。
- VGA パススルー は完全仮想化ゲストにのみ対応しています。準仮想化ゲストの場合、サポートされていません。
- 複数の VM ゲスト でグラフィックカードを共用することはできません。いずれか 1 台のゲストでのみ使用することができます。

VGA パススルー を有効化するには、完全仮想化ゲスト側の設定ファイル内に、下記の内容を追加します:

```
gfx_passthru=1
pci=['yy:zz.n']
```

ここで、`yy:zz.n` には VGA グラフィックアダプタの PCI デバイス番号を指定します。この値は Dom0 で `lspci -v` を実行することで、取得することができます。

24.5.4 トラブルシューティング

状況によっては VM ゲスト のインストール中に問題が発生することがあります。本章では、既知の問題とその解決方法について説明しています。

起動時にシステムがハングアップする

ソフトウエア I/O 変換バッファは、起動時の早い段階で low メモリに大きなチャンクを確保しようとします。もしもバッファのサイズ以上にメモリを要求してしまうと、通常は起動処理がハングアップしてしまいます。この問題に該当しているかどうかを調べるには、コンソールの 10 番に切り替えて、下記のようなメッセージが現れていないかどうかを確認してください:

```
kernel: PCI-DMA: Out of SW-IOMMU space for 32768 bytes at device 000:01:02.0
```

このような場合、`swiotlb` のサイズを増やす必要があります。Dom0 のコマンドラインに対して、`swiotlb=値` の形式で値を設定してください。なお、`値` には slab エントリ数を指定します。なお、この値を増やしたり減らしたりして、マシンに対する最適値を判断してください。



注記: PV ゲストでの swiotlb について

PV ゲスト内で PCI デバイスを動作させるには、DMA アクセスに対して `swiotlb=force` というカーネルパラメータを指定する必要があります。IOMMU や `swiotlb` に関するさらに詳しい情報については、`kernel-source` パッケージ内にある `boot-options.txt` ファイルをお読みください。

24.5.5 さらなる情報

PCI パススルー に関する様々な情報がインターネット上に公開されています (いずれも英語です):

- https://wiki.xenproject.org/wiki/VTd_HowTo
- https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices/
- https://support.amd.com/TechDocs/48882_IOMMU.pdf

24.6 USB パススルー

ホスト側の USB デバイスをゲスト側にパススルーする方法としては、2 種類のものがあります。1 つは擬似 USB デバイスコントローラを介する方法、もう 1 つは PVUSB を使用する方法です。

24.6.1 USB デバイスの識別方法

USB デバイスを VM ゲスト にパススルーする前に、まずは VM ホストサーバ 内でデバイスを識別する必要があります。ホスト側で `lsusb` コマンドを実行し、USB デバイスの一覧を表示してください:

```
# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0461:4d15 Primax Electronics, Ltd Dell Optical Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

パススルーしたいデバイスを、`製造元 ID:デバイス ID` の形式のデバイスタグ (上記の例では (0461:4d15)) もしくは `バス番号.デバイス番号` の形式のバスアドレス (上記の例では (2.3)) でメモしておきます。なお、冒頭の 0 については削除しておいてください。これは、0 を取っておかないと、`xl` コマンドが 8 進数として解釈してしまうためです。

24.6.2 擬似 USB デバイス

擬似 USB デバイスを使用する場合、デバイスのモデル (QEMU) はゲストに対して擬似 USB コントローラを提供します。USB デバイスは Dom0 から制御され、USB のコマンドは VM ゲスト からホスト側の USB デバイスに送信される際、変換が行われます。この方式は、完全仮想化環境の仮想マシン (HVM) でのみ使用することができます。

擬似 USB ハブを有効化するには、`usb=1` オプションを設定します。あとは設定ファイル内で、接続されているデバイスの中からデバイスを選択します。このとき、`host:USBID` のように指定します。たとえば下記のようになります:

```
usb=1
usbdevice=['tablet','host:2.3','host:0424:460']
```

24.6.3 準仮想化 PVUSB

PVUSB は Dom0 から仮想化ゲストに USB デバイスをパススルーするための新しく高性能な方式です。PVUSB では、USB デバイスを追加するにあたって、下記の 2 種類の方式を使用することができます:

- 仮想マシンの作成時に設定ファイルで指定する方法
- VM 動作中のホットプラグ接続による方法

PVUSB では準仮想化によるフロントエンド／バックエンドインターフェイスを使用します。PVUSB では USB 1.1 と USB 2.0 に対応するほか、PV と HVM の両方のゲストで使用することができます。PVUSB を使用するには、ゲスト OS 側に usbfront を、dom0 もしくは qemu 内の USB バックエンド内で usbback を使用します。openSUSE Leap では、qemu に USB バックエンドが付属しています。

Xen バージョン 4.7 およびそれ以降のバージョンで `xl` の PVUSB サポートとホットプラグサポートが追加されています。

設定ファイル内で指定する場合は、`usbctrl` と `usbdev` で USB のコントローラとホストデバイスを指定します。たとえば HVM ゲストの場合、下記のようになります:

```
usbctrl=['type=qusb,version=2,ports=4', 'type=qusb,version=1,ports=4', ]
```

```
usbdev=['hostbus=2, hostaddr=1, controller=0,port=1', ]
```



注記

HVM ゲストのコントローラに対しては、`type=qusb` を指定する点が重要です。

PVUSB デバイスのホットプラグを管理するには、`usbctrl-attach`，`usbctrl-detach`，`usb-list`，`usbdev-attach`，`usb-detach` の各サブコマンドを使用します。たとえば下記のようになります:

USB 1.1 に対応し、8 ポートを持つ USB コントローラを作成するには:

```
# xl usbctrl-attach test_vm version=1 ports=8 type=qusb
```

仮想マシン内にある最初のコントローラ:ポートを検出し、そこにバス番号:デバイス番号が 2:3 である USB デバイスを接続するには (`controller` (コントローラ) と `port` (ポート) を指定することもできます):

```
# xl usbdev-attach test_vm hostbus=2 hostaddr=3
```

仮想マシン内で利用可能な全ての USB コントローラとデバイスを表示するには:

```
# xl usb-list test_vm
Devid  Type   BE  state usb-ver ports
0      qusb   0   1     1       8
  Port 1: Bus 002 Device 003
  Port 2:
  Port 3:
  Port 4:
  Port 5:
  Port 6:
  Port 7:
  Port 8:
```

コントローラ 0 ポート 1 に接続されている USB デバイスを取り外すには:

```
# xl usbdev-detach test_vm 0 1
```

`dev_id` で表される USB コントローラと、それに接続されている全ての USB デバイスを取り外すには:

```
# xl usbctrl-detach test_vm dev_id
```

詳しくは https://wiki.xenproject.org/wiki/Xen_USB_Passthrough (英語) をお読みください。

25 仮想ネットワーク

改訂履歴

2024-06-27

一方の VM ゲスト システムから他の VM ゲスト システムやその他のネットワークと通信を行う必要がしばしば発生します。VM ゲスト 側でのネットワークインターフェイスは独立したデバイスドライバで動作しているものであり、Dom0 側にそれらの仮想イーサネットデバイスに対応するネットワークインターフェイスが存在していることになります。このインターフェイスは Dom0 内で動作する仮想ネットワークにアクセスするために作られています。このようなブリッジ型の仮想ネットワークは openSUSE Leap のシステム設定内に完全に統合されていて、YaST からも設定を行うことができるようになっています。

Xen の VM ホストサーバ をインストールすると、通常のネットワーク設定の際にブリッジ型のネットワーク設定を行うよう提案が表示されます。ユーザ側ではインストール時に設定を変更することができるほか、必要に応じてカスタマイズを行うこともできます。

また、システムのインストールを行った後からでも、YaST 内に用意された **ハイパーバイザとツールのインストール** を使用することで、必要に応じて Xen VM ホストサーバ をインストールすることができます。このモジュールは仮想マシンを動作させるためのシステム側の準備を行う仕組みで、既定のネットワークブリッジの構築も行うことができます。

`rpm` や `zypper` などを利用して手作業で Xen VM ホストサーバ に必要なパッケージをインストールした場合、残りのシステム設定は管理者が手作業で行うか、YaST を利用して行う必要があります。

Xen 側で提供されているネットワークスクリプトは、openSUSE Leap の既定では使用されません。これらは念のため用意されているだけであり、無効化されています。openSUSE Leap の Xen で使用するネットワークの設定は、通常のネットワークインターフェイス設定と同様に、YaST のシステム設定で行います。

ネットワークブリッジの管理方法に関する一般的な情報については、8.1.1項「ネットワークブリッジ」をお読みください。

25.1 ゲストシステム向けのネットワークデバイス

Xen ハイパーバイザでは、VM ゲスト 側に提供することのできるネットワークインターフェイスの種類が複数用意されています。可能であれば準仮想化ネットワークインターフェイスを使用するのが推奨されます。これにより、最低限のシステム要件で最大の伝送速度を得ることができます。また、それぞれの VM ゲスト に対して最大で 8 個までのネットワークインターフェイスを設定することができます。

準仮想化ハードウエアに対応していないシステムの場合、準仮想化ネットワークインターフェイスを使用することができません。この場合、システムをネットワークに接続するには、完全仮想化環境での擬似ネットワークインターフェイスを使用します。下記の擬似ネットワークインターフェイスを使用することができます:

- Realtek 8139 (PCI) (既定の擬似ネットワークカードです)
- AMD PCnet32 (PCI)
- NE2000 (PCI)
- NE2000 (ISA)
- Intel e100 (PCI)
- Intel e1000 およびその派生ハードウエア (e1000-82540em, e1000-82544gc, e1000-82545em) (PCI)

これらのネットワークインターフェイスはいずれも、ソフトウエアインターフェイスです。それぞれのネットワークインターフェイスには、他のものと重複しないユニークな MAC アドレスを設定しなければなりませんが、Xensource に割り当てられた MAC アドレスの中から使用することになります。



ヒント: 仮想ネットワークインターフェイスと MAC アドレスについて

仮想環境内の MAC アドレスの設定は、既定で乱数を使用して設定され、00:16:3E:xx:xx:xx のような値になります。通常は MAC アドレスの範囲はそれなりに大きいため、重複する可能性はほとんどありませんが、非常に大量のマシンを配置するような場合や、乱数による MAC アドレスの割り当てで発生しうる問題を完全に回避したい場合は、手作業で MAC アドレスを設定してください。

デバッグやシステム管理を行う目的で、ゲスト内で動作しているイーサネットデバイスが Dom0 内の仮想インターフェイスのどれに接続されているのかを知っておくと便利です。この情報は Dom0 内の名前付けルールから判別することができます。仮想インターフェイスの名前は、`vif<ドメイン番号>.<インターフェイス番号>` の形式になります。

たとえば ID=5 の VM ゲスト の 3 番目のインターフェイス (eth2) の Dom0 でのデバイス名は、`vif5.2` になります。利用可能な全てのインターフェイスを表示したい場合は、`ip a` コマンドを実行してください。

デバイスの命名では、そのインターフェイスがどのブリッジに接続されているのかを知ることができません。しかしながら、この情報は Dom0 内で管理されています。ブリッジに接続されているインターフェイスを知りたい場合は、`bridge link` コマンドを実行してください。出力は下記のようになります:

```
> sudo bridge link
```

```
2: eth0 state DOWN : <NO-CARRIER,BROADCAST,MULTICAST,SLAVE,UP> mtu 1500 master br0
3: eth1 state UP : <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 master br1
```

上記の例では `br0` , `br1` , `br2` の 3 つのブリッジが設定されていて、そのうち `br0` と `br1` には実際のイーサネットデバイス (`eth0` と `eth1`) が接続されていることになります。

25.2 Xen でのホストベースルーティング

Xen では Dom0 での制御によってホストベースのルーティングを設定することができます。ですが、残念なことに YaST では十分にサポートされておらず、設定ファイルを多数手作業で編集する必要があります。そのため、この設定は知識のあるシステム管理者が行う必要があります。

下記の設定では、固定の IP アドレスを利用した設定を行っています。この設定では VM ゲスト と VM ホストサーバ との間で互いに IP アドレスを知っておく必要があることから、DHCP 環境での使用には対応していません。

ルーティング型のゲストを構築するのに最も簡単な方法は、ブリッジ型のネットワークをルーティング型のネットワークに変更することです。下記の手順でも、VM ゲスト に対してあらかじめブリッジ型のネットワークが構成されているものとしています。また下記の説明では、VM ホストサーバ が earth という名前で 192.168.1.20 というアドレス、そして VM ゲスト が alice という名前で 192.168.1.21 というアドレスであるものとします。

手順 25.1: ルーティング型 IPV4 VM ゲスト の設定

1. alice がシャットダウンされていることを確認します。`xl` コマンドを実行してシャットダウンとその確認を行ってください。
2. VM ホストサーバ earth のネットワーク設定を準備します:
    a. トラフィックをルーティングするためのホットプラグ型インターフェイスを作成します。これを実現するには、`/etc/sysconfig/network/ifcfg-alice.0` ファイルを作成して、下記の内容を記述します:

    ```
    NAME="Xen guest alice"
    BOOTPROTO="static"
    STARTMODE="hotplug"
    ```

    b. IP 転送を有効化します:
        i. YaST を起動して、[ネットワークの設定] › [ルーティング] を開きます。
        ii. [ルーティング] タブ内にある [IPv4 転送を有効にする] と [IPv6 転送を有効にする] をそれぞれ選択します。

iii. 設定を適用して YaST を終了します。

c. `firewalld` に対して下記のとおり設定を適用します:

- public ゾーン内のデバイスに alice.0 を追加します:

```
> sudo firewall-cmd --zone=public --add-interface=alice.0
```

- 転送すべきアドレスをファイアウオールに設定します:

```
> sudo firewall-cmd --zone=public \
--add-forward-
port=port=80:proto=tcp:toport=80:toaddr="192.168.1.21/32,0/0"
```

- 設定を恒久的に保存します:

```
> sudo firewall-cmd --runtime-to-permanent
```

d. alice のインターフェイスに対してスタティックルートを追加します。これを行うには、下記の内容を `/etc/sysconfig/network/routes` の末尾に追加します:

```
192.168.1.21  -  -  alice.0
```

e. VM ホストサーバ が接続しているスイッチやルータに対して、ルーティングインターフェイスであることを認識させるため、earth 側で `proxy_arp` を有効化します。下記の内容を `/etc/sysctl.conf` に追加します:

```
net.ipv4.conf.default.proxy_arp = 1
net.ipv4.conf.all.proxy_arp = 1
```

f. 下記のコマンドを実行して変更点を適用します:

```
> sudo systemctl restart systemd-sysctl wicked
```

3. あとは 26.1項「xl: Xen 管理ツール」で説明している内容に従って alice 向けの vif インターフェイス設定を変更し、VM ゲスト の Xen 設定を行います。処理中に生成したテキストファイルに対して、下記の変更を行います:

a. 下記の内容を削除します:

```
bridge=br0
```

b. 下記の内容を追加します:

```
vifname=vifalice.0
```

もしくは

```
vifname=vifalice.0=emu
```

(完全仮想化ゲストの場合)

c. インターフェイスを設定する際のスクリプトを下記のように変更します:

```
script=/etc/xen/scripts/vif-route-ifup
```

d. 新しい設定を適用し、VM ゲスト を起動します。

4. 残りの設定作業は VM ゲスト 内から実施します。

a. `xl console` 仮想マシン名 のように入力して実行し、VM ゲスト のコンソールを開いてログインします。

b. ゲスト側の IP アドレスが 192.168.1.21 になっていることを確認します。

c. VM ホストサーバ に対して VM ゲスト へのホストルートとデフォルトゲートウエイを設定します。具体的には、下記の内容を `/etc/sysconfig/network/routes` に追加します:

```
192.168.1.20 - - eth0
default 192.168.1.20 - -
```

5. 最後に VM ゲスト からインターネット側へのアクセスを確認するとともに、ネットワーク側から VM ゲスト 側へのアクセスも確認します。

25.3 マスカレード型ネットワーク設定

マスカレード型のネットワークの設定はルーティング型の設定に似ていますが、proxy_arp の設定が不要となるほか、いくつかのファイアウオールルールの設定が異なります。dolly という名前のゲストが 192.168.100.1 というアドレスで存在し、外部インターフェイスと接続されているブリッジが `br0` である場合、下記のように実施します。設定を簡単にするため、インストール済みのゲストに対してマスカレード型ネットワークを設定するものとします:

手順 25.2: マスカレード型 IPV4 VM ゲストの設定

1. `xl shutdown` 仮想マシン名 のように入力して実行し、VM ゲスト をシャットダウンします。

2. VM ホストサーバ 側でのネットワーク設定を行います:

   a. トラフィックをルーティングするためのホットプラグ型インターフェイスを作成します。これを実現するには、`/etc/sysconfig/network/ifcfg-dolly.0` ファイルを作成して、下記の内容を記述します:

   ```
   NAME="Xen guest dolly"
   BOOTPROTO="static"
   STARTMODE="hotplug"
   ```

   b. `/etc/sysconfig/SuSEfirewall2` ファイルを編集し、下記の設定を追加します:

      - FW_DEV_DMZ のデバイス内に dolly.0 を追加します:

      ```
      FW_DEV_DMZ="dolly.0"
      ```

      - ファイアウオールでルーティングを有効化します:

      ```
      FW_ROUTE="yes"
      ```

      - ファイアウオールでマスカレードを有効化します:

      ```
      FW_MASQUERADE="yes"
      ```

      - ファイアウオールに対して、マスカレード処理を行うように指定します:

      ```
      FW_MASQ_NETS="192.168.100.1/32"
      ```

      - マスカレードの例外からネットワークを削除します:

      ```
      FW_NOMASQ_NETS=""
      ```

      - 最後にファイアウオールを再起動します:

      ```
      > sudo systemctl restart SuSEfirewall2
      ```

   c. dolly のインターフェイスに対してスタティックルートを追加します。これを行うには、下記の内容を `/etc/sysconfig/network/routes` の末尾に追加します:

   ```
   192.168.100.1 - - dolly.0
   ```

   d. これまでに行った変更を反映させます:

   ```
   > sudo systemctl restart wicked
   ```

3. あとは VM ゲスト の Xen 設定を行います。

a. あとは 26.1項「xl: Xen 管理ツール」で説明している内容に従って alice 向けの vif インターフェイス設定を変更します。

b. 下記の内容を削除します:

```
bridge=br0
```

c. 下記の内容を追加します:

```
vifname=vifdolly.0
```

d. インターフェイスを設定する際のスクリプトを下記のように変更します:

```
script=/etc/xen/scripts/vif-route-ifup
```

e. 新しい設定を適用し、VM ゲスト を起動します。

4. 残りの設定作業は VM ゲスト 内から行います:

a. `xl console` 仮想マシン名 のように入力して実行し、VM ゲスト のコンソールを開いてログインします。

b. ゲスト側の IP アドレスが 192.168.100.1 に設定されていることを確認します。

c. VM ホストサーバ に対して VM ゲスト へのホストルートとデフォルトゲートウエイを設定します。具体的には、下記の内容を `/etc/sysconfig/network/routes` に追加します:

```
192.168.1.20 - - eth0
default 192.168.1.20 - -
```

5. 最後に VM ゲスト からインターネット側へのアクセスを確認します。

25.4 特殊な設定

Xen ではさまざまなネットワーク設定を行うことができます。下記の設定はいずれも、既定では有効化されていないものです:

25.4.1 仮想ネットワーク内での帯域制限

Xen では仮想化ゲストに対して、ブリッジにアクセスする際の許可帯域を設定することができます。この設定を行うには、まず 26.1項「xl: Xen 管理ツール」に示している手順に従って、VM ゲスト の設定を変更する必要があります。

設定ファイル内で、仮想ブリッジに接続されているデバイスを探します。設定は下記のようになっているはずです:

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0' ]
```

最大帯域を設定するには、この設定の中に `rate` という項目を追加します。たとえば下記のようになります:

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0,rate=100Mb/s' ]
```

なお、帯域の指定は `Mb/s` (メガビット毎秒) もしくは `MB/s` (メガバイト毎秒) の単位で指定することができます。上記の例では、仮想インターフェイスに対する最大帯域を 100 メガビット毎秒に指定しています。既定では仮想ブリッジに対するゲストの帯域制限はありません。

また、帯域計算の粒度として時間を指定して調整を図ることもできます。たとえば下記のようになります:

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0,rate=100Mb/s@20ms' ]
```

25.4.2 ネットワークトラフィックの監視

特定のインターフェイスに対してトラフィックを監視するには、`iftop` というプログラムを利用して、端末内でトラフィックの監視を行うとよいでしょう。

Xen の VM ホストサーバ を動作させている場合、監視対象のインターフェイスを指定して動作させる必要があります。ここで指定するインターフェイスは、物理ネットワークが接続されているブリッジデバイスとなります (例: `br0`) 。そのため、インターフェイス名はシステムによって異なることになります。全ての物理インターフェイスを監視したい場合は、`root` で端末を起動して、下記のコマンドを実行してください:

```
iftop -i br0
```

特定の VM ゲスト のインターフェイスのトラフィックのみを監視したい場合は、仮想インターフェイスを指定して実行します。たとえばドメイン ID が 5 の 1 つめのイーサネットデバイスを監視したい場合は、下記のようなコマンドになります:

```
ftop -i vif5.0
```

`iftop` を終了させるには `Q` キーを押します。オプションや用途に関する詳細は、`man 8 iftop` で表示されるマニュアルページをお読みください。

26 仮想環境の管理

改訂履歴

2024-06-27

推奨される方式である `libvirt` ライブラリによる管理 (パートII「libvirt を利用した仮想マシンの管理」) とは別に、コマンドラインから Xen 専用のツールである `xl` を使用することで、Xen のゲストドメインを管理することもできます。

26.1 xl: Xen 管理ツール

`xl` コマンドは Xen ゲストドメインを管理するためのツールです。このプログラムは `xen-tools` パッケージ内に含まれています。`xl` コマンドは LibXenlight ライブラリをベースにして作られていますので、ドメインの作成や一覧表示、一時停止やシャットダウンなど、一般的なゲスト管理を行うことができます。通常は `xl` コマンドを実行するのに `root` の権限が必要となります。



注記

`xl` コマンドは設定ファイルを指定しますが、動作中のゲストドメインのみを管理することができます。対象のゲストドメインが動作中ではない場合、`xl` コマンドによる管理を行うことができません。



ヒント

従来の `xm` コマンドで実現できていた、一般ユーザからのゲストドメイン管理を行いたい場合は、`libvirt` 側に用意されている `virsh` と `virt-manager` の各ツールをお使いください。詳しくは パートII「libvirt を利用した仮想マシンの管理」をお読みください。

`xl` を動作させるには、`xenstored` サービスと `xenconsoled` サービスを開始しておく必要があります。下記を実行して開始しておいてください:

```
> systemctl start xencommons
```

"¥n ¥n"



ヒント: ホストドメイン内での xenbr0 ネットワークブリッジの設定について

最も一般的なネットワーク設定では、ホストドメイン側に `xenbr0` という名前のブリッジを作成して、ゲストドメインのネットワークを動作させます。

`xl` コマンドの基本構造は下記のとおりです:

```
xl <サブコマンド> [オプション] ドメイン_ID
```

ここで、`<サブコマンド>` には実行すべき xl のサブコマンドを、`ドメイン_ID` にはドメインに対して割り当てられた ID か、仮想マシンの名前を、そして `[オプション]` にはサブコマンドごとに固有のオプション類を指定します。

利用可能な全ての `xl` のサブコマンドの一覧を表示するには、`xl help` コマンドを実行してください。このほか、各サブコマンドに対して詳細なヘルプも表示することができます。この場合は、サブコマンドのオプションとして `--help` を指定してください。また、サブコマンドに関する詳細は、`xl` のマニュアルページにも記載があります。

たとえば `xl list --help` のように入力して実行すると、list コマンドに対して指定可能な全てのオプションが表示されます。ちなみに、`xl list` コマンドは、全ての仮想マシンの状態を一覧表示するためのコマンドです。

```
> sudo xl list
Name                                        ID    Mem VCPUs        State   Time(s)
Domain-0                                     0    457     2       r-----   2712.9
sles12                                       7    512     1       -b----     16.3
opensuse                                          512     1                  12.9
```

[State] の列には、対象のマシンが動作しているかどうかと、その現在の状態が示されています。最もよく表示されるのが `r` (running; 動作中) と `b` (blocked; ブロック中) (IO 処理待ち、もしくは何もすべきことが無く、待機中) です。状態 (State) フラグの一覧に関する詳細は、`man 1 xl` のマニュアルページをお読みください。

その他の `xl` サブコマンドには、下記のようなものがあります:

- `xl create`: 指定した設定ファイルから仮想マシンを作成します。
- `xl reboot` : 仮想マシンを再起動します。
- `xl destroy`: 仮想マシンを即時に停止します。
- `xl block-list`: 仮想マシンに接続されている全ての仮想マシンを表示します。

26.1.1 ゲストドメインの設定ファイル

ゲストドメイン (仮想マシン) を動作させるにあたって、`xl` ではゲストドメイン向けの設定ファイルが必要となります。設定ファイルを保存しておくための既定のディレクトリは `/etc/xen/` です。

ドメインの設定ファイルは純粋なテキスト形式です。ここには `キー = 値` の形式で、さまざまな設定が記述されます。キーによっては必須のものもあるほか、一般的で任意のゲストに適用することができるものや、特定の種類のゲストにのみ適用できるもの (たとえば完全仮想化環境専用の設定など) があります。値は `"文字列"` のようにして、単一もしくは二重引用符で括ることもできるほか、数値やブール値、もしくは `[ 値_1, 値_2, ... ]` のように括弧で複数の値をまとめることもできます。

例 26.1: SLED 12 のゲストドメイン設定ファイル例: `/etc/xen/sled12.cfg`

```
name= "sled12"
builder = "hvm"
vncviewer = 1
memory = 512
disk = [ '/var/lib/xen/images/sled12.raw,,hda', '/dev/cdrom,,hdc,cdrom' ]
vif = [ 'mac=00:16:3e:5f:48:e4,model=rtl8139,bridge=br0' ]
boot = "n"
```

上記のドメインを起動するには、`xl create /etc/xen/sled12.cfg` を実行します。

26.2 ゲストドメインの自動起動

ホスト側のシステムが起動した際にゲストドメインを自動的に起動したい場合は、下記の手順を実施します:

1. ドメインの設定ファイルを作成していない場合は、`/etc/xen/` ディレクトリ内に作成します。たとえば `/etc/xen/domain_name.cfg` のようになります。
2. `auto/` サブディレクトリ内に、ゲストドメインの設定ファイルに対するシンボリックリンクを作成します。

   ```
   > sudo ln -s /etc/xen/domain_name.cfg /etc/xen/auto/domain_name.cfg
   ```

3. このように設定することで、システムの起動時に `domain_name.cfg` で設定されたゲストドメインが開始するようになります。

26.3 イベントアクション

ゲスト側の設定ファイルでは、さまざまなイベントに対応するアクションを設定することができます。たとえばゲスト側の電源が落とされた際、その後にドメインを再起動させたい場合は、設定ファイル内に下記を記述します:

```
on_poweroff="restart"
```

ゲストドメインで利用できるイベントには、下記のようなものがあります:

イベントの一覧

on_poweroff
: ドメインをシャットダウンした際に取るべきアクションを指定します。

on_reboot
: ドメインが再起動を要求するような理由コード付きのシャットダウンを受け取った際に、取るべきアクションを指定します。

on_watchdog
: Xen のウオッチドッグタイムアウトにより、ドメインをシャットダウンする際に取るべきアクションを指定します。

on_crash
: ドメインがクラッシュした際に取るべきアクションを指定します。

これらのイベントに対して、下記のアクションを指定することができます:

関連アクションの一覧

destroy
: ドメインを強制停止します。

restart
: ドメインを強制停止したあと、同じ設定のまま新しいドメインを作成します。

rename-restart
: ドメインを強制停止したあと名前を変更し、元の名前で新しいドメインを作成します。

preserve
: ドメインを維持します。ここから何らかの調査を行うことができます。調査が終わったら、`xl destroy` で停止を行うことができます。

coredump-destroy
: `/var/xen/dump/名前` 内にドメインのコアダンプを書き込んだあと、ドメインを強制停止します。

coredump-restart

`/var/xen/dump/名前` 内にドメインのコアダンプを書き込んだあと、ドメインを再起動します。

26.4 タイムスタンプカウンタ (TSC)

Time Stamp Counter (TSC) をゲストドメインの設定ファイル (詳しくは 26.1.1項「ゲストドメインの設定ファイル」をお読みください) 内に設定することができます。

`tsc_mode` の設定を行うことで、rdtsc インストラクションをネイティブで動作させる (高速に動作するものの、TSC を厳格に処理するアプリケーションが正しく動作しなくなってしまうことがあります) か、もしくは擬似的に動作させる (常に正しく動作しますが、性能が劣化します) かを制御することができます。

`tsc_mode=0` (既定値)

性能をできるだけ劣化させず、かつ正確性を維持する方式です。詳しくは https://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt (英語) をお読みください。

`tsc_mode=1` (常に擬似的に動作させる)

TSC を厳格に処理するアプリケーションが存在していて、かつ最大限の性能劣化を許容できるシステムである場合に指定する方式です。

`tsc_mode=2` (擬似的に動作させない)

TSC を厳格に処理するアプリケーションが存在せず、最大限の性能を発揮させる必要のあるシステムである場合に指定する方式です。

`tsc_mode=3` (PVRDTSCP)

頻繁に TSC を使用するアプリケーションが存在する場合、準仮想化 (修正済み) で動作させることによって、正確性と高性能の両方を実現することができます。未修正のアプリケーションについては、TSC に対する耐性が必須となります。

背景となる情報について、詳しくは https://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt をお読みください。

26.5 仮想マシンの保存

手順 26.1: 仮想マシンに対する現在状態の保存

1. まずは保存したい仮想マシンが動作中であることを確認します。

2. ホスト側の環境で下記を実行します:

```
> sudo xl save ID 状態ファイル
```

ここで `ID` には保存対象の仮想マシンの ID を、`状態ファイル` にはメモリの状態を保存するためのファイルを指定します。既定では、このようにして保存を行った場合、対象のドメインは停止されますが、`-c` を指定することで動作させたままの状態にすることもできます。

26.6 仮想マシンの再開

手順 26.2: 仮想マシンに対する保存状態の再開

1. まずは対象の仮想マシンが開始されていないことを確認します。
2. ホスト側の環境で下記を実行します:

```
> sudo xl restore 状態ファイル
```

ここで、`状態ファイル` には保存時に使用したファイルのファイル名 (パス) を指定します。既定では再開を行うと動作中の状態に戻りますが、`-p` を指定することで、一時停止中のままにしておくこともできます。

26.7 仮想マシンの状態

仮想マシンの状態は `xl list` コマンドで表示することができます。このとき、状態は下記に示す 1 文字の略記で表されます。

- `r` (running (動作中)): 仮想マシンが動作していて、割り当てられたリソースを使用していることを示しています。
- `b` (blocked (ブロック中)): 仮想マシンのプロセッサが動作しておらず、かつ動作できない状態であることを示しています。I/O の処理待ち、もしくは停止処理中であることを表しています。
- `p` (paused (一時停止中)): 仮想マシンが一時停止中であることを表しています。ハイパーバイザとの通信は行われていませんが、メモリなどのリソースは使用したままの状態になります。
- `s` (shutdown (シャットダウン中)): ゲスト側のオペレーティングシステムでシャットダウンや再起動、サスペンド処理などを行っていて、仮想マシンが停止されつつある状態であることを表しています。

- `c` (crashed (クラッシュ済み)): 仮想マシンがクラッシュしていて、動作していないことを表しています。
- `d` (dying (死亡中)): 仮想マシンがシャットダウン中であるか、クラッシュ中であることを表しています。

27 Xen 内でのブロックデバイス

改訂履歴

2024-06-27

27.1 物理ストレージから仮想ディスクへのマッピング

ドメインの設定ファイル内にある Xen ドメインのディスク設定は非常に直感的で、下記のように記述します:

```
disk = [ 'format=raw,vdev=hdc,access=ro,devtype=cdrom,target=/root/image.iso' ]
```

上記では `/root/image.iso` というファイルをベースにしたディスク型のブロックデバイスを定義しています。ゲスト内では、このディスクは `hdc` として表示され、読み込み専用 (`ro`) に設定されています。デバイスの種類は `cdrom` で、`raw` 形式を使用する設定になっています。

下記の設定例は上記と同じ設定ですが、よりシンプルなカンマ区切りの書式を使用しています:

```
disk = [ '/root/image.iso,raw,hdc,ro,cdrom' ]
```

同じ行内に複数のディスク定義を設定したい場合は、それぞれをカンマ区切りで指定してください。パラメータを指定しない場合、既定値が指定されているものとみなされます:

```
disk = [ '/root/image.iso,raw,hdc,ro,cdrom','/dev/vg/guest-volume,,hda','...' ]
```

パラメータの一覧

target
: ソースとなるブロックデバイスやディスクイメージのファイルパスを指定します。

format
: イメージファイルの形式を指定します。既定値は `raw` です。

vdev
: ゲスト側での仮想デバイスを指定します。設定できる値には hd[x], xvd[x], sd[x] などがあります。詳しくは `/usr/share/doc/packages/xen/misc/vbd-interface.txt` (英語) ファイルをお読みください。このパラメータは必須です。

access
: ゲスト側でのブロックデバイスを読み込み専用とするか、読み書きできるものとするかを指定します。設定可能な値は `ro` または `r` (読み込み専用), `rw` または `w` (読み書き可能) から指定します。既定値は `devtype=cdrom` の場合は `ro` に、それ以外のデバイスの場合は `rw` になります。

devtype
: 仮想デバイスの種類を指定します。設定可能な値は `cdrom` です。

backendtype
: 使用するバックエンド実装を指定します。設定可能な値は `phy`，`tap`，`qdisk` のいずれかです。通常は、バックエンドの種類を自動設定するため、このオプションを指定する必要はありません。

script
: `target` が通常のホストパスではなく、実行可能なプログラムで解釈される情報である場合に指定します。ここで指定したスクリプトファイルが絶対パスではない場合、`/etc/xen/scripts` 内にあるものとして扱われます。これらのスクリプトは通常、`block-<スクリプト名>` と呼ばれます。

仮想ディスクの指定方法に関する詳細は、`/usr/share/doc/packages/xen/misc/xl-disk-configuration.txt` (英語) をお読みください。

27.2 ネットワークストレージから仮想ディスクへのマッピング

ローカルのディスクイメージをマッピングする (詳しくは 27.1項「物理ストレージから仮想ディスクへのマッピング」をお読みください) 場合と同様に、ネットワークディスクを仮想ディスクとしてマッピングすることもできます。

下記の例では、複数の Ceph モニタと cephx 認証が有効化された RBD (RADOS Block Device) ディスクに対して、マッピングを設定しています:

```
disk = [ 'vdev=hdc, backendtype=qdisk, \
target=rbd:libvirt-pool/new-libvirt-image:\
id=libvirt:key=AQDsPWtW8JoXJBAAyLPQe7MhCC+JPkI3QuhaAw==:auth_supported=cephx;none:\
mon_host=137.65.135.205\\:6789;137.65.135.206\\:6789;137.65.135.207\\:6789' ]
```

NBD (Network Block Device) のディスクマッピング例は下記のとおりです:

```
disk = [ 'vdev=hdc, backendtype=qdisk, target=nbd:151.155.144.82:5555' ]
```

27.3 ファイルとして存在する仮想ディスクとループバックデバイス

仮想マシンが動作している場合、ファイルとして存在する仮想ディスクは、ホスト内でループバックデバイスを使用します。既定では、ホスト側では 64 個までのループバックデバイスを使用することができます。

ホスト側でそれ以上のファイルベースの仮想ディスクを使用する場合は、ホスト側の `/etc/modprobe.conf.local` ファイル内に下記のオプションを追加して、同時に利用可能なループバック数を増やしておく必要があります。

```
options loop max_loop=x
```

ここで、`x` には使用したいループバックデバイスの最大数を指定します。

設定した内容は、モジュールの再読み込み時に反映されます。



ヒント

`rmmod loop` および `modprobe loop` を実行することで、モジュールの読み込み解除と再読み込みを行うことができます。なお、`rmmod` がうまく動作しない場合、既存のループバックデバイスの使用を全て解除するか、もしくはコンピュータを再起動してください。

27.4 ブロックデバイスのサイズ変更

VM ゲスト システムに新しいブロックデバイスを追加することもできますが、場合によっては既存のブロックデバイスのサイズを増やしたほうが都合のよい場合があります。VM ゲスト の作成の際にそのような拡張を考慮しておきたい場合は、下記のような設定を行っておくことをお勧めします:

- サイズを増やすことのできるブロックデバイスを使用するようにしてください。具体的には LVM デバイスやファイルベースのイメージがそれにあたります。
- VM ゲスト 内部ではデバイスのパーティションを作成せずに使用するものとし、メインとなるデバイスを直接マウントしてお使いください。たとえば `/dev/xvdb` に対してパーティションを作成する代わりに、`/dev/xvdb` を直接マウントしてください。
- サイズを変更することのできるファイルシステムを使用してください。たとえば ext3 などのファイルシステムでは、サイズを変更するためにいくつかの機能を無効化しなければならないこともあります。また、オンラインでマウントしたままサイズを変更することができるファイルシステムとして、

`XFS` があります。`xfs_growfs` コマンドを使用することで、ブロックデバイス側のサイズ変更を行ったあと、ファイルシステム側のサイズ変更を行うことができます。`XFS` の詳細について、詳しくは `man 8 xfs_growfs` をお読みください。

VM ゲスト に対して割り当てられている LVM デバイスをサイズ変更する場合、新しいサイズは自動的に VM ゲスト に伝達されます。新しいブロックデバイスのサイズを知らせるために必要なアクションはありません。

ファイルシステムのイメージを使用する場合、ループバックデバイスを使用してイメージファイルをゲストに割り当てます。イメージのサイズ変更と VM ゲスト 側でのサイズ情報の伝達について、詳しくは 29.2項「スパースイメージファイルとディスク領域」をお読みください。

27.5 高度なストレージ管理向けのスクリプト

ソフトウエア RAID セット上に構築されてた LVM 環境や LVM 環境上に構築されたソフトウエア RAID セットなど、`dmmd` (「device mapper—multi disk」) で提供されるディスク環境のような高度なストレージ環境の管理を支援するため、いくつかのスクリプトが提供されています。これらのスクリプトは `xen-tools` パッケージ内に含まれていて、インストール後には `/etc/xen/scripts` ディレクトリ内に存在します:

- `block-dmmd`
- `block-drbd-probe`
- `block-npiv`

これらのスクリプトを使用すると、外部のコマンドがゲストに対してブロックデバイスを提供する前に、そのブロックデバイスに対して何らかの処理や一連のアクションを実行することができるようになります。

これらのスクリプトは従来、`script=` のディスク設定書式を利用して `xl` や `libxl` で使用できていたものです。これらはディスクの `<source>` 要素内でブロックスクリプトのベース名を指定することで、libvirt でも使用できるようになっています。たとえば下記のようになります:

```
<source dev='dmmd:md;/dev/md0;lvm;/dev/vgxen/lv-vm01'/>
```

28 仮想化: オプション設定

改訂履歴

2024-06-27

本章では高度な管理作業のほか、最先端の仮想化ソリューションを実装したいユーザ向けの設定オプションについて説明しています。ただし、本章で説明しているオプションや作業はいずれも善意で提供されているものであり、Novell 社のサポート範囲内であることを示すものではありません。

28.1 仮想 CD リーダ

仮想 CD リーダは、仮想マシンの作成時に追加することができるほか、既存の仮想マシンに対しても追加を行うことができます。また、仮想 CD リーダは物理 CD/DVD をベースにして動作させることができるほか、ISO イメージをベースにすることもできます。なお、仮想 CD リーダは、準仮想化の場合と完全仮想化の場合で異なる動作になります。

28.1.1 準仮想化マシン内での仮想 CD リーダ

準仮想化型の仮想マシンでは、仮想 CD リーダと仮想ディスクを足して最大で 100 個までの設定を行うことができます。準仮想化マシンでは、仮想 CD リーダは読み込み専用の仮想ディスクとして CD を表示します。仮想 CD リーダは、CD のデータ書き込み用に使用することはできません。

準仮想化型のマシンで CD へのアクセスが終わったら、仮想マシンから仮想 CD リーダを削除しておくことをお勧めします。

準仮想化ゲストでは、`devtype=cdrom` というデバイス種類の設定を行うことができます。この設定を行うことで実際の CD リーダの動作を擬似するほか、メディアの交換にも対応するようになります。また、CD リーダのトレイを開くためのイジェクトコマンドも使用できるようになります。

28.1.2 完全仮想化マシン内での仮想 CD リーダ

完全仮想化マシンの場合、仮想 CD リーダと仮想ディスクを足して最大で 4 個までの設定を行うことができます。完全仮想化マシンの仮想 CD リーダは、通常の (物理的な) CD リーダと全く同じ扱いをすることができます。

ホストコンピュータ側の物理 CD リーダ (例: `/dev/cdrom`) にメディアを挿入すると、物理 CD リーダをベースにした仮想 CD リーダが設定されている全ての仮想マシンから、挿入したメディアを読み込むことができるようになります。なお、オペレーティングシステム側に自動マウントの機能があれば、CD は自動的にファイルシステム内に現れるようになります。仮想 CD リーダは CD のデータ書き込み用に使用することはできません。読み込み専用デバイスとして設定されます。

28.1.3 仮想 CD リーダの追加

仮想 CD リーダは、ホストコンピュータに接続された CD リーダをベースにすることができるほか、ISO イメージファイルをベースにすることもできます。

1. 仮想マシンが動作中であり、オペレーティングシステムの起動が完了していることを確認します。
2. まずは物理 CD リーダにメディアを挿入するか、もしくは必要な ISO イメージを Dom0 内に準備します。
3. VM ゲスト で新しく未使用のブロックデバイスを選択します。たとえば `/dev/xvdb` などになります。
4. ゲストに割り当てるものが物理 CD リーダなのか ISO イメージなのかを選択します。
5. 物理 CD リーダを使用する場合は、下記のコマンドを VM ゲスト に対して実行し、CD リーダの割り当てを行います。下記の例は、ゲストの名前が alice である場合の例となります:

```
> sudo xl block-attach alice target=/dev/sr0,vdev=xvdb,access=ro
```

6. イメージファイルを割り当てる場合は、下記のコマンドを実行します:

```
> sudo xl block-attach alice target=/path/to/file.iso,vdev=xvdb,access=ro
```

7. これで `/dev/xvdb` などのブロックデバイスを仮想マシンに追加することができました。
8. 仮想マシンが Linux である場合は、下記のように実行します:

    a. 仮想マシン内で端末を開いて `fdisk -l` を実行し、デバイスが正しく割り当てられていることを確認します。`ls /sys/block` を実行することでも、仮想マシンに存在するすべきのディスクを表示することができます。
    CD は仮想マシン側で仮想ディスクとして認識されます。そのため、下記のようなデバイス名になります:

```
/dev/xvdb
```

b. 下記のようなコマンドを入力して実行することで、物理メディアもしくは ISO イメージを読み込むことができます。

```
> sudo mount -o ro /dev/xvdb /mnt
```

上記を実行すると、`/mnt` というマウントポイントにマウントされます。
すると、指定したマウントポイントにアクセスすることで、CD もしくは ISO イメージの内容にアクセスすることができるようになります。

9. 仮想マシンが Windows である場合は、仮想マシンを再起動します。
マイコンピュータ 内に仮想 CD リーダが現れていることを確認します。

28.1.4 仮想 CD リーダの削除

1. 仮想マシンが動作中であり、オペレーティングシステムの起動が完了していることを確認します。
2. 仮想 CD リーダがマウントされている場合は、まずは仮想マシン内でマウントを解除します。
3. ホスト側で `xl block-list alice` のように入力して実行することで、ゲスト側のブロックデバイスの一覧を表示することができます。
4. ゲストから仮想デバイスを削除するには、`xl block-detach alice ブロックデバイス_ID` のように入力して実行します。これがうまくいかない場合は、`-f` オプションを指定して強制的な取り外しをお試しください。
5. メディアを取り出すには、物理的な取り出しボタンを押してください。

28.2 リモートアクセス方式

ラックマウント型のサーバなど、ビデオモニタやキーボード、マウスが接続されていないコンピュータは、しばしば **ヘッドレス** 環境と呼ばれ、これらを VM ホストサーバ とする場合は、リモート管理技術を設定しておく必要があります。
一般的なシステム設定としては、下記のようなものがあります:

X Window Server によるグラフィカルデスクトップ
: GNOME などのグラフィカルデスクトップを仮想マシンホストにインストールしている場合、VNC ビューアなどのリモートビューアを使用してログインすることができるほか、`tigervnc` や `virt-viewer` などを使用することで、リモートのゲスト環境にログインすることもできます。

テキストのみ

`ssh` コマンドをリモートのコンピュータで実行することで、仮想マシンホストのシェルにログインして、テキストベースのコンソールを使用することができます。あとは `xl` コマンドで仮想マシンの管理を行うことができるほか、`virt-install` コマンドで新しい仮想マシンを作成することもできます。

28.3 VNC ビューア

VNC ビューアを使用することで、動作中のゲストに対してグラフィカルな方法でアクセスすることができます。Dom0 からアクセスする際に使用する (ローカルアクセスやオンボックスアクセスと呼びます) ことができるほか、リモートのコンピュータからアクセスする際にも使用することができます。

VM ゲスト のディスプレイを表示したい場合は、接続先は VM ホストサーバ の IP アドレスになります。仮想マシンが動作中の場合、ホスト側の VNC サーバがポート番号を割り当てて、VNC ビューアからのアクセスを待ち受けます。割り当てられるポート番号は、仮想マシンの起動時における最小のポート番号になります。なお、ポート番号は仮想マシンの動作中にのみ有効であり、シャットダウンを行ってしまうと、ポート番号は他の仮想マシンに割り当てることができるよう、解放されます。

たとえば 1, 2, 4, 5 の各ポートが仮想マシン向けに割り当てられている場合、新しく起動する仮想マシンに対しては、VNC サーバが 3 を割り当てることになります。同じ仮想マシンであっても、次回の起動時に 3 が使用中であった場合は、また別のポート番号を割り当てることになります。

リモートのコンピュータから VNC サーバに接続できるようにするには、ファイアウオール側で VM ゲスト の使用するポートをできる限り多く空けておく必要があります。VNC では 5900 番以降を使用する仕組みであることから、たとえば 10 台の VM ゲスト を動作させるような場合、TCP ポートの 5900 から 5910 までを空けておいてください。

VM ホストサーバ 内のローカルで仮想マシンの VNC ディスプレイにアクセスするには、下記のいずれかのコマンドを実行します:

- `vncviewer ::590#`
- `vncviewer :#`

ここで、`#` には仮想マシンに割り当てられた VNC ポート番号を指定します。

VM ホストサーバ 以外のマシンから VM ゲスト にアクセスする場合は、下記の書式でコマンドを作成して実行します:

```
> vncviewer 192.168.1.20::590#
```

上記は、VM ホストサーバ のアドレスが 192.168.1.20 である場合の例となります。

28.3.1 仮想マシンに対する VNC ビューアのポート番号設定

VNC サーバの既定では、利用可能な最小のポート番号を自動的に割り当てる動作を行いますが、仮想マシンに対して固定の VNC ポート番号を割り当てることもできます。

特定の VM ゲスト が使用するポートを固定したい場合は、仮想マシンの xl 設定ファイルを編集して、下記のような箇所にある `vnclisten` の値を必要に応じて変更してください。下記の例では 2 を指定していますが、これは 5900 のベース値が自動的に足されて、5902 として扱われます:

```
vfb = [ 'vnc=1,vnclisten="localhost:2"' ]
```

ゲストドメインの xl 設定の編集に関する詳細は、26.1項「xl: Xen 管理ツール」をお読みください。



ヒント

動的に最も小さい VNC ポート番号を割り当てる仮想マシンが存在する場合は、それらとの重複を避けるため、大きめの値を設定しておいてください。

28.3.2 VNC ビューアの代替としての SDL の使用

仮想マシンのディスプレイを仮想マシンホストのコンソール自身から行う場合 (ローカルアクセスやオンボックスアクセスと呼ばれます) 、VNC ではなく SDL を使用することもできます。VNC はネットワーク経由でデスクトップを表示するには高速なプロトコルですが、同じコンピュータ内でデスクトップにアクセスするだけであれば、SDL のほうがより高速に処理することができます。

VNC ではなく SDL を既定値として使用する場合は、仮想マシンの xl 設定を下記のように変更してください。変更方法に関する詳細は 26.1項「xl: Xen 管理ツール」をお読みください。

```
vfb = [ 'sdl=1' ]
```

なお、VNC ビューアからのアクセスとは異なり、SDL のウインドウを閉じてしまうと、仮想マシンそのものも終了してしまうことに注意してください。

28.4 仮想キーボード

仮想マシンが起動すると、ホスト側では仮想マシンの設定内にある `keymap` の内容に従って、仮想キーボードを作成します。`keymap` の設定が存在しない場合、仮想マシンは既定値である英語 (US) キーボードを作成します。

仮想マシンの現在のキーボード設定を確認するには、下記のコマンドを Dom0 で実行します:

```
> xl list -l VM_名 | grep keymap
```

ゲストの仮想キーボードを設定するには、下記のような内容を設定します:

```
vfb = [ 'keymap="ja"' ]
```

対応するキーボードレイアウトの一覧を表示したい場合は、`man 5 xl.cfg` で表示されるマニュアルページ内にある、`Keymaps` セクションをご覧ください。

28.5 CPU リソースの占有

Xen 内では、Dom0 や VM ゲスト が性能を確保する目的で、使用する CPU コア数やコア番号を指定することができます。Dom0 の性能は、ディスクやネットワークのドライバがここで動作することから、システム全体に対して重要な設定となります。I/O に負荷が集中するゲストの場合、Dom0 の CPU サイクルを多く消費することになります。その一方、VM ゲスト 側の性能についても、必要な処理を完了するためのリソースとなりますので、こちらも重要となります。

28.5.1 Dom0 側の設定

Dom0 に対して CPU リソースを占有できるようにすることで、VM ゲスト 側から届く I/O 要求を Dom0 内で容易に処理できるようになることから、性能を向上させる結果になります。逆に Dom0 が CPU リソースを占有できないと、性能が落ちてしまうだけでなく、VM ゲスト が正しく動作しない場合もあります。

CPU リソースの占有を設定するには、3 つの手順が必要となります。まずは Xen の起動時のコマンドライン設定、次に Dom0 の VCPU 設定、そして最後に VM ゲスト に対する CPU 関連の設定になります。

1. まずは Xen の起動時のコマンドラインに対して、`dom0_max_vcpus=X` を追加します。具体的には、`/etc/default/grub` に下記の行を追加します:

   ```
   GRUB_CMDLINE_XEN="dom0_max_vcpus=X"
   ```

   `/etc/default/grub` 内に既に `GRUB_CMDLINE_XEN` がある場合は、その行に `dom0_max_vcpus=X` を追加してください。
   なお、`X` には Dom0 が占有する VCPU 数を指定します。

2. あとは下記のコマンドを実行して、GRUB 2 の設定ファイルを更新します:

   ```
   > sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

3. コンピュータを再起動して、設定を反映させます。

4. 次に Dom0 の VCPU のそれぞれに対して、物理プロセッサへのバインド (「ピン」とも呼びます) を設定します。

```
> sudo xl vcpu-pin Domain-0 0 0
xl vcpu-pin Domain-0 1 1
```

最初の行は、Dom0 の VCPU 番号 0 を物理プロセッサ番号 0 に割り当てる設定、次の行は Dom0 の VCPU 番号 1 を物理プロセッサ番号 1 に割り当てる設定です。

5. 最後に、全ての VM ゲスト の設定を変更して、Dom0 が使用する物理プロセッサを使用しないようにします。たとえば 8-CPU のシステムであれば、下記のような内容を全ての VM ゲスト の設定ファイル内に追加します:

```
cpus="2-8"
```

"¥n ¥n"

28.5.2 VM ゲスト 側の設定

仮想マシン側に対しても、特定の CPU リソースを占有するように設定する必要がある場合があります。既定では、仮想マシンは利用可能な CPU コアを全て使用します。なお、十分な数の物理プロセッサを占有させ、他の VM ゲスト に対して使用させないようにすることで、性能を改善することができます。たとえば 8 CPU コアのマシンに対して、仮想マシンがそのうちの 2 個を占有するような場合、下記のような設定になります:

```
vcpus=2
cpus="2,3"
```

上記の例では `2` 個のプロセッサを VM ゲスト に専用に割り当てています。使用するコアは 3 番目と 4 番目 (設定ファイル上では 0 から数えるため、`2` と `3` になっています) です。さらに多くの物理プロセッサを割り当てる必要がある場合は、`cpus="2-8"` のような書式で指定してください。

「alice」という名前のゲストに対して、CPU の割り当てをホットプラグ形式で変更したい場合は、対応する Dom0 で下記のコマンドを実行してください:

```
> sudo xl vcpu-set alice 2
> sudo xl vcpu-pin alice 0 2
> sudo xl vcpu-pin alice 1 3
```

上記の例では、ゲストに対して 2 個の物理プロセッサを占有するよう設定しています。それぞれ VCPU 0 が物理プロセッサ 2 を、VCPU 1 が物理プロセッサ 3 を使用します。あとは下記のようにすることで、割り当てを確認することができます:

```
> sudo xl vcpu-list alice
```

```
Name                                ID VCPUs   CPU State   Time(s) CPU Affinity
alice                               4     0     2   -b-       1.9 2-3
alice                               4     1     3   -b-       2.8 2-3
```

28.6 HVM 機能

Xen 環境では、完全仮想化環境のドメインにのみ提供される機能が存在しています。これらは頻繁に使用するものではありませんが、環境によっては設定する必要があるかもしれません。

28.6.1 起動時のブートデバイス指定

物理ハードウエアと同様に、VM ゲスト を通常使用するものとは異なる起動デバイスを使用する必要が生じることがあります。完全仮想化型のマシンであれば、ドメインの xl 設定ファイル内に `boot` パラメータを指定することで、起動に使用するデバイスを設定することができます:

```
boot = 起動デバイス
```

ここで、*起動デバイス* には、`c` であればハードディスクを、`d` であれば CD-ROM を、`n` であればネットワーク (PXE) 起動を指定することになります。複数を指定した場合は、並んだ順に起動を試す動作になります。たとえば下記のようになります:

```
boot = dc
```

上記の例は、CD-ROM からの起動を試し、それがうまくいかない場合はハードディスクから起動しようとする設定になります。

28.6.2 ゲスト向けの CPUID 変更

一方の VM ホストサーバ から他方の VM ホストサーバ に VM ゲスト を移行できるようにするには、両方の VM ホストサーバ システムのいずれでも使用できる CPU 機能のみを提供するように設定する必要があります。つまり、両方の VM ホストサーバ で実際に搭載されている CPU が異なる場合、VM ゲスト の起動時に、共通しない機能を隠蔽しておく必要があることになります。これにより、ホストを跨いで VM ゲスト を移行できることになります。完全仮想化環境の場合、これは `cpuid` を設定することで実現することができます。

現在搭載されている CPU の情報を取得するには、`/proc/cpuinfo` ファイルをご覧ください。ここには現在の CPU に関する重要な情報が全て記載されています。

CPU の再定義を行うには、まず CPU の製造元が提供する CPUID の定義を参照しておく必要があります。これらはそれぞれ下記の箇所で公開されています (いずれも英語です):

Intel
: https://www.intel.com/Assets/PDF/appnote/241618.pdf

```
cpuid = "host,tm=0,sse3=0"
```

書式はカンマ区切りで `キー=値` の形式で指定します。また、最初は必ず "host" と記述します。いくつかのキーには数値を指定しますが、残りのほとんどのキーには、機能ビットを操作するための 1 文字を記述します。CPUID のキーについて、詳しくは `man 5 xl.cfg` をお読みください。また、下記の値を指定することで、対応するビットの指定を変えることができます:

1
: 対応するビットを `1` に強制します

0
: 対応するビットを `0` に強制します

x
: 既定のポリシーの値を使用します

k
: ホスト側の値をそのまま使用します

s
: `k` と同様ですが、移行後も値を保持するようにします



ヒント

なお、ビットは右から左に表記し、かつ `0` から始めます。

28.6.3 PCI-IRQ 数の拡張

Dom0 や VM ゲスト に提供する PCI-IRQ の数を増やす必要がある場合、Xen カーネルのコマンドラインでそれを行うことができます。`extra_guest_irqs=` `DOMU_IRGS,DOM0_IRGS` の形式で指定してください。最初の値である `DOMU_IRGS` は、全ての VM ゲスト に対する値を、2 つめの値である

`DOM0_IRGS` (カンマ区切り) は、Dom0 に対する値とになります。なお、VM ゲスト の設定を変更しても、Dom0 には何も影響しませんし、逆もまた然りです。たとえば Dom0 側のみを変更したい場合は下記のように指定します:

```
extra_guest_irqs=,512
```

28.7 仮想 CPU のスケジューリング

Xen ハイパーバイザでは、全ての物理 CPU にまたがる形で個別に仮想 CPU のスケジュール処理を行います。各コアに複数のスレッドが搭載された新しい CPU の場合、複数の仮想 CPU が同一コア内の別スレッドとして動作することがありますので、これによって仮想 CPU の処理性能に影響がある場合があります。たとえば一方のスレッド内で動作する仮想 CPU が重い処理をしている場合、もう一方のスレッド内で動作する仮想 CPU の性能が大きく落ち込んでしまうことになります。また、これらの仮想 CPU が別々のゲストシステムを動作させている場合は、ゲストシステム側にも影響があることになります。この場合の性能劣化は状況によって大きく異なり、単純にゲストシステム側に割り当てられる処理時間が削減されるだけであることもありますし、最悪の場合は サイドチャネル攻撃 という結果をもたらすこともあります。

このような問題に対しては、スケジューリング粒度 の設定を行うことをお勧めします。Xen の起動パラメータとして下記のような値を設定してください:

```
sched-gran=粒度
```

`粒度` の箇所には下記のいずれかを指定します:

cpu
: Xen ハイパーバイザでの標準的なスケジュール設定です。1 つの物理 CPU コア内で複数のゲストの仮想 CPU を動作させます。こちらが既定値になります。

core
: 1 つの物理 CPU コアに対して、常に 1 つの仮想 CPU のコア 1 つを割り当てます。同じ物理 CPU コア内で複数の仮想 CPU のコアを動作させることはありません。そのため、動作させるべき仮想 CPU が存在するような状況でも、いくつかの物理コアが待機状態に置かれることがあります。性能への影響は、ゲストシステム内で実際に動作させる処理内容によって異なります。なお、比較的負荷の高い状況下では、1 つのコア内で 1 つの処理のみを動作させるハイパースレッディングの無効化 (詳しくは https://xenbits.xen.org/docs/unstable/misc/xen-command-line.html#smt-x86 にある `smt` オプションの説明をお読みください) より、性能劣化は穏やかになります。

socket
: さらに粒度を高めて、CPU ソケット単位で処理を分けるようにします。

29 管理作業

改訂履歴

2024-06-27

29.1 ブートローダプログラム

ブートローダには仮想化ソフトウエアの起動と実行に関する制御が含まれています。YaST を使用するか、もしくは直接設定ファイルを編集することで、ブートローダプログラムの設定を変更することができます。

YaST のブートローダプログラムは［YaST］›［システム］›［ブートローダ］にあります。［ブートローダのオプション］タブを選択して、Xen カーネルを含むものを［既定のブートセクション］で選択してください。

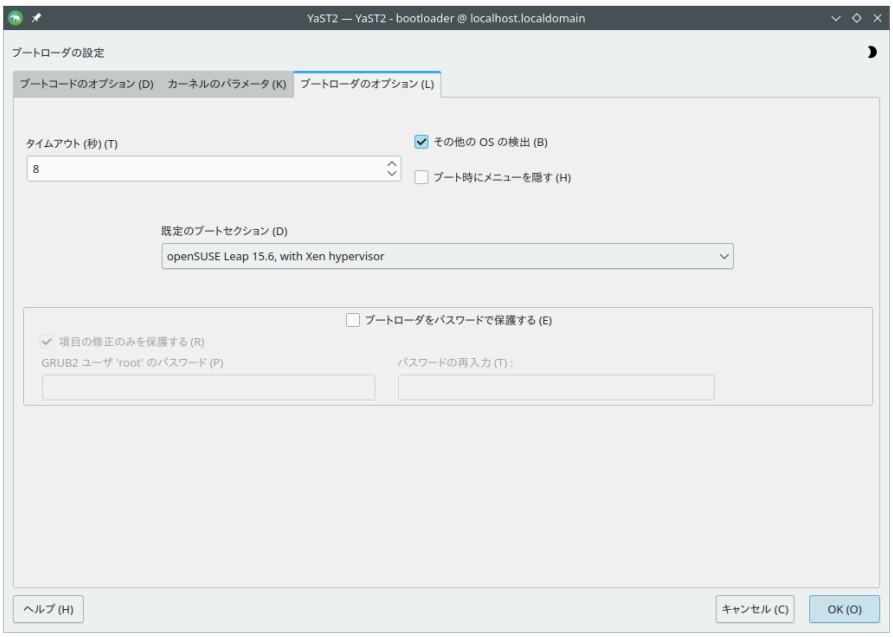


図 29.1: ブートローダの設定

設定が終わったら［OK］を押します。これでホストの次回の起動時に Xen が読み込まれ、Xen 仮想化環境を使用できるようになります。

ブートローダプログラムを使用することで、様々な機能を調整することができます。具体的には下記のようなものがあります:

- カーネルのコマンドラインパラメータの指定
- カーネルイメージと初期 RAM ディスクの指定
- 特定のハイパーバイザの選択
- ハイパーバイザに対する追加パラメータの指定。詳しくは https://xenbits.xen.org/docs/unstable/misc/xen-command-line.html (英語) をお読みください。

仮想化環境のカスタマイズに際しては、`/etc/default/grub` ファイルを編集してください。このファイルに対して、`GRUB_CMDLINE_XEN="<パラメータ>"` のような行を追加します。なお、編集後は忘れずに `grub2-mkconfig -o /boot/grub2/grub.cfg` を実行してください。

29.2 スパースイメージファイルとディスク領域

ホスト側の物理ディスクの容量がいっぱいになり空き容量が無くなると、スパース (疎な) イメージファイルをベースとした仮想ディスクを使用している仮想マシンは、ディスクへの書き込みができなくなります。これにより、I/O エラーが発生することになります。

このような状況に陥ってしまった場合、まずは物理ディスク側にある不要なファイルを削除するなどして容量を空けてください。その後、仮想マシン側のファイルシステムを再マウントして、ファイルシステムを書き込み可能な状態に戻してください。

スパースイメージファイルが実際に使用している容量を調べるには、`du -h <イメージファイル名>` を実行します。

スパースイメージファイルの容量を増やすには、まずファイルサイズを増やしてから、ファイルシステム側のサイズを増やしてください。



警告: サイズ変更を実施する前にバックアップを採取しておく必要性について

パーティションのサイズを変更する場合もスパースファイルのサイズを変更する場合も、処理の失敗に備えて必ずバックアップを採取しておいてください。バックアップ無しに作業を行ってはなりません。

イメージファイルのサイズ変更はオンラインで実施することができます。つまり、VM ゲスト が動作している間でも実行することができます。スパースイメージファイルのサイズを拡張するには、下記のコマンドを入力して実行します:

```
> sudo dd if=/dev/zero of=<イメージファイル> count=0 bs=1M seek=<サイズ_(メガバイト単位)>
```

たとえば `/var/lib/xen/images/sles/disk0` というファイルを 16GB まで拡張させたい場合は、下記を実行します:

```
> sudo dd if=/dev/zero of=/var/lib/xen/images/sles/disk0 count=0 bs=1M seek=16000
```



注記: 非スパースイメージの拡張

非スパースファイルをベースにしたイメージファイルの場合であっても、イメージファイルのサイズを拡張することができます。ただし、この場合は現時点でのイメージサイズを正確に知っておかなければなりません。現時点でのイメージサイズを seek パラメータに設定して、拡張するサイズを count サイズで指定します:

```
> sudo dd if=/dev/zero of=/var/lib/xen/images/sles/disk0 seek=8000 bs=1M count=2000
```

なお、seek の値を間違えてはなりません。誤った値を指定してしまうと、データの一部を消してしまうことになります。

サイズ変更の作業時に VM ゲスト が動作中であった場合、VM ゲスト 側にイメージを提供しているループバックデバイス側でもサイズ変更を行う必要があります。まずは下記のコマンドを実行して、現在使用しているループバックデバイスを検出します:

```
> sudo losetup -j /var/lib/xen/images/sles/disk0
```

あとはループバックデバイスのサイズを変更します。たとえばループバックデバイスが `/dev/loop0` である場合は、下記のようなコマンドになります:

```
> sudo losetup -c /dev/loop0
```

あとはゲストシステム内でブロックデバイスのサイズを確認します。具体的には `fdisk -l /dev/xvdb` のようなコマンドを入力して実行し、確認を行います。なお、デバイス名はお使いの環境に合わせて変更してください。

スパースファイル内にあるファイルシステム側の変更については、使用しているファイルシステムの種類によって異なりますので、対応するツールをお使いのうえ変更を行ってください。

29.3 Xen VM ゲスト システムの移行

Xen では、ほとんどサービスの停止を伴うことなく VM ゲスト のシステムを一方の VM ホストサーバ から他方の VM ホストサーバ に移行することができます。これはたとえば、負荷の重い VM ゲスト をより強力なハードウエアの搭載された VM ホストサーバ に移行したり、負荷に余裕のある VM ホストサーバ に移行したりする際に使用します。また、何らかの理由で VM ホストサーバ を停止させる必要が発生した場合、その中で動作している VM ゲスト を他の VM ホストサーバ に移行して、サービスの停止を回避するためにも使用します。ここには 2 種類の理由しか記述していませんが、実際にはさまざまな理由で移行が必要となることがあります。

移行を開始する前に、まずは VM ホストサーバ に関するいくつかの事前要件を考慮する必要があります:

- 移行に関わる全ての VM ホストサーバ のシステムで、同じような CPU を使用する必要があります。動作周波数などは異なっていてもかまいませんが、同一のファミリの CPU を使用しておく必要があります。CPU に関する詳細は、`cat /proc/cpuinfo` を実行すると取得することができます。ホスト CPU の機能比較に関する詳細については、29.3.1項「CPU 機能の検出」をお読みください。
- 特定のゲストシステムが使用している全てのリソースは、移行に関わる全ての VM ホストサーバ から使用できなければなりません。具体的には、使用している全てのブロックデバイスは、両方の VM ホストサーバ システム内に存在していなければなりません。
- 移行処理に関わる VM ホストサーバ が異なるサブネット内に存在している場合、ゲストに対して DHCP リレーを提供する必要があります。なお、ゲスト側で固定のネットワーク設定を使用している場合、ネットワークの設定果て座興で修正する必要があります。
- `PCI パススルー` などの特殊機能を使用していると、問題が発生する場合があります。異なる VM ホストサーバ 間で VM ゲスト を移行する可能性がある場合は、これらの機能を使用しないでください。
- 移行を高速に行うには、ネットワークも高速化してください。可能であればギガビットイーサネットで高速なスイッチを使用してください。また、VLAN を使用することで、衝突を防ぐことができます。

29.3.1 CPU 機能の検出

`cpuid` および `xen_maskcalc.py` のツールを使用することで、移行元と移行先の VM ゲスト に存在する CPU の機能を比較することができます。これらのコマンドを使用することで、ゲストの移行が正しく完了するかどうかを事前によりよく予測することができるようになります。

1. 現在動作中の Dom0 と移行先の VM ゲスト 内で `cpuid -1r` を実行して出力を比較します。たとえば下記のようになります:

```
tux@vm_host1 > sudo cpuid -1r > vm_host1.txt
tux@vm_host2 > sudo cpuid -1r > vm_host2.txt
tux@vm_host3 > sudo cpuid -1r > vm_host3.txt
```

2. 出力されたテキストファイルを `xen_maskcalc.py` スクリプトがインストールされたホストに集めます。

3. あとは出力された全てのテキストファイルに対して `xen_maskcalc.py` スクリプトを実行します:

```
> sudo xen_maskcalc.py vm_host1.txt vm_host2.txt vm_host3.txt
cpuid = [
    "0x00000001:ecx=x00xxxxxx0xxxxxxxxx00xxxxxxxxxx",
    "0x00000007,0x00:ebx=xxxxxxxxxxxxxxxxxx00x0000x0x0x00"
]
```

4. あとは `cpuid=[...]` 内に出力された内容を、移行先のゲストの `domU.cfg` の `xl` 設定内に貼り付けるか、もしくは `libvirt` の XML 設定内に設定します。

5. あとは移行元のゲストを起動し直して、CPU の機能を 一部無効化 した形で実行します。これにより、全てのホスト内に存在する CPU 機能だけを使用して動作させることができるようになります。

29.3.1.1 さらなる情報

`cpuid` に関する詳細については、https://etallen.com/cpuid.html (英語) をお読みください。

CPU マスク計算プログラムの最新版をダウンロードしたい場合は、https://github.com/twizted/xen_maskcalc をご覧ください。

29.3.2 移行のためのブロックデバイスの準備

VM ゲスト 側で必要となるブロックデバイスは、移行に関わる全ての VM ホストサーバ システムからアクセスできなければなりません。これは移行を行う VM ゲスト システムのルートファイルシステムを、共有型ストレージ内に配置することで実現することができます。一般的には下記のようなものがあります:

- `iSCSI`: 複数のシステムに対して、同時に同一のブロックデバイスにアクセスする機能を提供することができます。
- `NFS`: 複数のシステムから容易にアクセスすることのできるルートファイルシステムとして、幅広く使用されています。詳しくは『リファレンス』、第22章「NFS によるファイル共有」をお読みください。
- `DRBD`: 2 台の VM ホストサーバ システム間でのみ使用することのできるシステムです。この仕組みを使用すると、ネットワークを介してデータが複製されることになりますので、データの保全性をさらに高めることができます。
- `SCSI`: ハードウエア側で共有が許可されていれば、同じディスクに対して複数のシステムがアクセスすることができます。
- `NPIV`: ファイバチャネルディスクを使用するための特殊なモードです。ただし、この場合は移行に関わる VM ホストサーバ が同じファイバチャネルのスイッチに接続されていなければなりません。NPIV に関する詳細は、27.1項「物理ストレージから仮想ディスクへのマッピング」をお読みください。一般的に、これは 4 Gbps もしくはそれ以上のファイバチャネル接続環境でのみ動作します。

29.3.3 VM ゲスト システムの移行

VM ゲスト システムの実際の移行作業は、下記のコマンドで行います:

```
> sudo xl migrate <ドメイン名> <ホスト名>
```

移行の速度はメモリのディスクへの書き込み速度と、新しい VM ホストサーバ への送信速度、そして新しい VM ホストサーバ での読み込み速度によって決まります。つまり、メモリの少ない VM ゲスト のほうが、メモリの多い VM ゲスト よりも素早く移行できることになります。

29.4 Xen の監視

多数の仮想化ゲストを定常的に管理している場合、動作している多数の VM ゲスト の正常性を監視する作業が欠かせません。Xen では、システムに関する情報を収集するためのさまざまなシステムツールを提供しています。



ヒント: VM ホストサーバ の監視

VM ホストサーバ の基本的な監視 (I/O および CPU) については、仮想マシンマネージャ 側で提供しています。詳しくは 10.7.1項「仮想マシンマネージャ を利用した監視」をお読みください。

29.4.1 `xentop` を利用した Xen の管理

Xen 仮想化環境で情報を収集するために使用する端末アプリケーションとして、`xentop` が用意されています。ただし、このツールを使用するにはサイズの大きな (文字数の多い) 端末が必要となります。サイズの小さな端末を使用してしまうと、表示が改行されて読みにくくなってしまいます。

`xentop` には監視の際の設定を変更するためのコマンドキーが用意されています。主なものは下記のとおりです:

D
: 画面の更新間隔を指定します。

N
: ネットワークの統計情報も表示するようにします。なお、標準的な設定のみが表示されます。また、ルーティング型ネットワークのような特殊な設定にも対応していません。

B
: 接続されているブロックデバイスと、その累積しよう回数を表示します。

`xentop` に関する詳細は、`man 1 xentop` で表示されるマニュアルページをお読みください。



ヒント: `virt-top`

libvirt にはハイパーバイザに依存しない `virt-top` というツールが提供されています。こちらも VM ゲスト の監視にはお勧めです。詳しくは 10.7.2項「`virt-top` を利用した監視」をお読みください。

29.4.2 追加のツール

稼働中の openSUSE システムを監視したりデバッグしたりするためのシステムツールは数多く存在しています。それらのうちの多くは『システム分析／チューニングガイド』、第2章「システム監視ユーティリティ」で説明を行っています。ここでは特に、仮想化環境を監視するためのツールについて説明しています:

ip

コマンドラインユーティリティである `ip` コマンドは、任意のネットワークインターフェイスの監視を行うことができるコマンドです。このコマンドは、ルーティング型のネットワークやマスカレード型のネットワークを構成しているような場合、特に有用です。たとえば `alice.0` という名前のネットワークインターフェイスを監視したい場合は、下記のコマンドを実行します:

```
> watch ip -s link show alice.0
```

bridge

標準的な設定では、全ての Xen VM ゲスト は仮想ネットワークブリッジに接続されます。`bridge` コマンドを実行すると、VM ゲスト システム内の仮想ネットワークアダプタが、どのブリッジに接続されているのかを知ることができます。この場合は、`bridge link` と入力して実行します。出力は下記のようになります:

```
2: eth0 state DOWN : <NO-CARRIER, ...,UP> mtu 1500 master br0
8: vnet0 state UNKNOWN : <BROADCAST, ...,LOWER_UP> mtu 1500 master virbr0 \
  state forwarding priority 32 cost 100
```

上記の出力では、2 種類の仮想ブリッジがシステム内に定義されています。一方は物理イーサネットデバイスである `eth0` に、もう一方は VLAN インターフェイスである `vnet0` に接続されています。

iptables-save

マスカレード型のネットワークを使用している場合や、イーサネットインターフェイスに対してファイアウォールの設定を行っている場合、現在のファイアウォールルールを確認しておく必要がある場合があります。

この場合、`iptables` コマンドを実行することで、さまざまなファイアウォール設定を一括で表示することができます。チェイン内の全てのルールを表示したい場合、もしくは全ての設定を出力したい場合は、`iptables-save` もしくは `iptables -S` を実行してください。

29.5 VM ゲスト システムに対するホスト情報の提供

標準的な Xen 環境では、VM ゲスト システムに対しては、VM ホストサーバ システムの一部の情報しか伝達されません。ゲスト側でさらに詳しい VM ホストサーバ の情報を提供する必要がある場合は、`vhostmd` を利用して、選択したゲストに情報提供を行ってください。お使いのシステムで `vhostmd` を動作させるには、下記の手順を行います:

1. まずは VM ホストサーバ に vhostmd パッケージをインストールします。
2. 設定内に `metric` セクションを追加または削除するには、`/etc/vhostmd/vhostmd.conf` ファイルを編集します。ただし、既定値のままでもかまいません。
3. 下記のコマンドを実行して、`vhostmd.conf` 設定ファイルの書式が正しいことを確認します:

   ```
   > cd /etc/vhostmd
   > xmllint --postvalid --noout vhostmd.conf
   ```

4. あとは `sudo systemctl start vhostmd` を実行して、vhostmd デーモンを開始します。vhostmd をシステムの起動時に開始するように設定したい場合は、下記を実行します:

   ```
   > sudo systemctl enable vhostmd
   ```

5. alice という名前の VM ゲスト に対して、`/dev/shm/vhostmd0` というイメージファイルを接続するため、下記のコマンドを実行します:

   ```
   > xl block-attach opensuse /dev/shm/vhostmd0,,xvdb,ro
   ```

6. VM ゲスト 側のシステムにログインします。
7. クライアント側のパッケージ `vm-dump-metrics` をインストールします。
8. `vm-dump-metrics` コマンドを実行します。結果をファイルに保存したい場合は、`-d <ファイル名>` オプションを追加してください。

`vm-dump-metrics` の出力は XML 形式で行われます。また、出力される内容は `/etc/vhostmd/metric.dtd` の DTD 定義に従って行われます。

さらに詳しい情報については、`man 8 vhostmd` で表示されるマニュアルページのほか、VM ホストサーバ システム側にある `/usr/share/doc/vhostmd/README` (英語) ファイルをお読みください。ゲスト側では、`man 1 vm-dump-metrics` のマニュアルページもお読みいただくことができます。

30 XenStore: ドメイン間で共有される設定データベース

改訂履歴

2023-12-22

本章では、XenStore に関する基本的な情報のほか、Xen 環境内での役割、XenStore で使用されるファイルのディレクトリ構造、そして XenStore のコマンドに関する説明をそれぞれ行っています。

30.1 概要

XenStore は設定や状態の情報を含むデータベースで、VM ゲストと Dom0 内で動作する管理ツールとの間で共有される仕組みでもあります。VM ゲストと管理ツールは、XenStore との間で情報を読み書きし、設定情報の伝達や状態の更新／変更などを行うことができます。XenStore データベースは Dom0 が管理するものであり、キーをベースにした読み書きなどのシンプルな構成になっています。VM ゲストと管理ツール側では、XenStore 内での項目を監視することで、変更の通知を受けることもできます。なお、`xenstored` デーモンは `xencommons` サービスで管理されます。

XenStore は Dom0 内で単一のデータベースファイル `/var/lib/xenstored/tdb` (`tdb` は Tree DataBase (ツリー構造型データベース) の略) として存在しています。

30.2 ファイルシステムインターフェイス

XenStore のデータベースの内容は `/proc` に似た仮想的なファイルシステム構造になっています (`/proc` に関する詳細は『システム分析／チューニングガイド』、第2章「システム監視ユーティリティ」、2.6項「/proc ファイルシステム」をお読みください) 。このファイルシステムのツリーには、`/vm` , `/local/domain` , `/tool` の 3 つの種類から構成されています。

- `/vm`: VM ゲストの設定に関する情報が保存されています。
- `/local/domain`: ローカルノード内の VM ゲストに関する情報が保存されています。
- `/tool`: さまざまなツールに関する一般的な情報が保存されています。



ヒント

それぞれの VM ゲストには 2 種類の ID 番号が設定されます。一方の Universal Unique IDntifier (UUID) は、VM ゲストを他のマシンに移行しても変更されることの無い値、もう一方の DOMain IDentifier (DOMID) は、動作中のインスタンスを一位に識別する番号で、こちらは一方から他方に移行を行うと、通常は値が変化することになります。

30.2.1 XenStore のコマンド

XenStore データベースのファイルシステム構造は、下記のコマンドを使用してアクセスすることができます:

`xenstore-ls`
: XenStore データベースのダンプを表示します。

`xenstore-read` `XenStore_エントリへのパス`
: 指定した XenStore の項目の値を読み取ります。

`xenstore-exists` `XenStore_のパス`
: 指定した XenStore のパスが存在しているかどうかを報告します。

`xenstore-list` `XenStore_のパス`
: 指定した XenStore パス以下の全ての子項目を表示します。

`xenstore-write` `XenStore_エントリへのパス`
: 指定した XenStore の項目の値を更新します。

`xenstore-rm` `XenStore_のパス`
: 指定した XenStore の項目またはディレクトリを削除します。

`xenstore-chmod` `XenStore_のパス` `モード`
: 指定した XenStore パスの読み込み／書き込み権限を更新します。

`xenstore-control`
: たとえば一貫性のチェックなど、`xenstored` バックエンドに対してコマンドを送信します。

30.2.2 /vm

`/vm` パスにはそれぞれの VM ゲスト の UUID が順に並べられ、その中には仮想 CPU 数や割り当てられているメモリ量などの設定情報が保存されています。つまり、それぞれの VM ゲスト に対して `/vm/<UUID>` というディレクトリが存在することになります。ディレクトリの内容を表示したい場合は、`xenstore-list` コマンドをお使いください。

```
> sudo xenstore-list /vm
00000000-0000-0000-0000-000000000000
9b30841b-43bc-2af9-2ed3-5a649f466d79-1
```

出力の冒頭は Dom0 そのもので、2 行目が実際の VM ゲスト になります。下記のように実行することで、VM ゲスト 内の項目を表示することができます:

```
> sudo xenstore-list /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1
image
rtc
device
pool_name
shadow_memory
uuid
on_reboot
start_time
on_poweroff
bootloader_args
on_crash
vcpus
vcpu_avail
bootloader
name
```

たとえば VM ゲスト に割り当てられている仮想 CPU 数など、項目の値を読み取りたい場合は、`xenstore-read` コマンドを使用します:

```
> sudo xenstore-read /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1/vcpus
1
```

`/vm/<UUID>` 以下の主な項目は下記のとおりです:

`uuid`
: VM ゲスト の UUID です。移行処理を行っても、この値は変更されません。

`on_reboot`
: VM ゲスト が再起動要求を受け取った場合、VM ゲスト のシャットダウンや再起動など、何を行うのかを指定する項目です。

`on_poweroff`
: VM ゲスト が停止要求を受け取った場合、VM ゲスト のシャットダウンや再起動など、何を行うのかを指定する項目です。

`on_crash`
: VM ゲスト がクラッシュした場合、VM ゲスト のシャットダウンや再起動など、何を行うのかを指定する項目です。

`vcpus`
: VM ゲスト に割り当てられた仮想 CPU 数を表します。

`vcpu_avail`
: VM ゲスト に対して提供されている仮想 CPU のビットマスクです。このビットマスクには `vcpus` の値と同じ数だけビットが設定されています。

`name`
: VM ゲスト の名前を表します。

通常の VM ゲスト (Dom0 以外) であれば、`/vm/<UUID>/image` のようなパスも存在します:

```
> sudo xenstore-list /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1/image
ostype
kernel
cmdline
ramdisk
dmargs
device-model
display
```

それぞれの項目の意味は下記のとおりです:

`ostype`
: VM ゲスト の OS の種類を表します。

`kernel`
: VM ゲスト で使用するカーネルの Dom0 内でのパスを表します。

`cmdline`
: VM ゲスト の起動時に使用するカーネルのコマンドラインを表します。

`ramdisk`
: VM ゲスト で使用する RAM ディスクの Dom0 内でのパスを表します。

`dmargs`

QEMU プロセスに対して渡されるパラメータを表します。`ps` コマンドで QEMU のプロセスを参照すると、`/vm/<UUID>/image/dmargs` と同じ内容が現れるはずです。

30.2.3 `/local/domain/<ドメイン_ID>`

ここでのパスは動作中のドメイン (VM ゲスト) の ID を使用し、動作中の VM ゲスト に対する情報を提供しています。なお、VM ゲスト の移行を行った場合、ドメイン ID は変化することに注意してください。下記のような項目が提供されます:

`vm`

この VM ゲスト に対応する `/vm` ディレクトリのパスを表します。

`on_reboot, on_poweroff, on_crash, name`

30.2.2項「`/vm`」での同名の項目と同じ意味を持ちます。

`domid`

VM ゲスト のドメイン ID を表します。

`cpu`

VM ゲスト 側でピン設定を行っている現在の CPU を表します。

`cpu_weight`

スケジューリングの目的で割り当てられた、VM ゲスト に対する重み値を表します。この値が大きければ大きいほど、物理 CPU が頻繁に割り当てられるようになります。

上述の項目以外にも、`/local/domain/<ドメイン_ID>` 内にはいくつかのサブディレクトリが存在しています。利用可能な全ての項目の意味を知りたい場合は、XenStore Reference (https://wiki.xen.org/wiki/XenStore_Reference) (英語) をお読みください。

`/local/domain/<ドメイン_ID>/memory`

メモリに関する情報が含まれています。なお、`/local/domain/<ドメイン_ID>/memory/target` には、VM ゲスト に対するターゲットのメモリサイズ (キロバイト単位) が含まれています。

`/local/domain/<ドメイン_ID>/console`

VM ゲスト が使用しているコンソールに関する情報が含まれています。

`/local/domain/<ドメイン_ID>/backend`

VM ゲスト が使用している全てのバックエンドデバイスの情報が含まれています。ここにはさらにサブディレクトリが存在する場合があります。

`/local/domain/<ドメイン_ID>/device`
: VM ゲスト のフロントエンドデバイスに関する情報が含まれています。

`/local/domain/<ドメイン_ID>/device-misc`
: デバイスに関するその他の情報が含まれています。

`/local/domain/<ドメイン_ID>/store`
: VM ゲスト のストア情報が含まれています。

31 Xen の高可用性仮想化ホストとしての使用

改訂履歴

2023-06-06

2 台の Xen ホストをフェイルオーバーシステムとして構築することで、サーバを別々に動作させる場合と比べて、さまざまなメリットが生まれます:

- 一方のサーバで障害が発生しても、サービスが止まらずに稼働させ続けることができます。
- 一般に 1 台のマシンのスペックを上げるよりも、2 台の同じスペックのマシンを用意したほうが安く上がります。
- 必要に応じて新しいサーバを継ぎ足していくと、管理面での手間が増えるだけです。
- サーバの使用方法を改善することで、システムの電源消費量に良い影響を与えることにもなります。

Xen ホスト間で移行を行うことのできる環境の構築については、29.3項「Xen VM ゲスト システムの移行」で説明しています。下記では、高可用性を実現するためのいくつかの方式を説明しています。

31.1 リモートストレージでの Xen HA 構成

Xen ではゲストシステムに対して、いくつかのリモートブロックデバイスを直接提供する機能を用意しています。これには iSCSI, NPIV, NBD があります。これらのうちの全てはライブマイグレーションでも使用することができます。ストレージシステムが既に配置されている場合、まずはネットワーク内で既に使用されているものと同じ種類のデバイスを使ってみてください。

ストレージシステムを直接使用することができないものの、NFS を介して必要な領域を提供できる機能を備えている場合は、NFS の共有内にイメージファイルを配置する方法もあります。NFS が全ての Xen ホストシステムからアクセスできる環境であれば、Xen ゲストのライブマイグレーションにも対応することができます。

新しいシステムを構築する場合の最初の考慮事項は、専用のストレージエリアネットワークを作成する必要があるかどうかです。これを考えるにあたっては、下記のような構成が考えられます:

表 31.1: XEN リモートストレージ

方式	複雑さ	コメント
イーサネット	低	この場合、ブロックデバイスのトラフィックはネットワークトラフィックと同じイーサネット内を

方式	複雑さ	コメント
		流れることになります。従って、ゲスト側の性能も制限されることになります。
ストレージ専用のイーサネット	中	ストレージ関係のトラフィックを専用のイーサネットインターフェイスから行うようにすることで、サーバ側でのボトルネックを解消することができます。この場合、専用のイーサネット側には独自の IP アドレス範囲を割り当てたり、ストレージ専用の VLAN を構築したりする必要性など、各種の考慮事項が存在します。
NPIV	高	NPIV はファイバチャネル接続の仮想化方式です。これには最低でも 4 Gbit/s のデータ転送速度を提供し、複雑なストレージシステムの構築にも対応しているアダプタで使用することができます。

通常は 1 Gbit/s 程度のイーサネットデバイスであれば、ハードディスクやストレージシステムの性能を生かすことができますが、さらに高速なストレージシステムをお使いの場合は、イーサネットデバイスの速度が不足してしまうこともありますので、ご注意ください。

31.2 ローカルストレージでの Xen HA 構成

スペースや予算上の都合から、Xen のホストシステムのローカルストレージを使用する必要がある場合があります。この場合にライブマイグレーションを実現するには、両方の Xen ホスト間でブロックデバイスを複製するよう構築する必要があります。このようなブロックデバイスの複製を行うソフトウエアとしては、Distributed Replicated Block Device (DRBD) と呼ばれるソフトウエアが存在します。

2 台の Xen ホスト間でブロックデバイスやファイルを複製するために DRBD を設定する必要がある場合、両方のホストでは等価なハードウエアを使用する必要があります。たとえば一方のホストのハードディスク速度が遅い場合、もう一方のホストもそれに引きずられてしまい、性能が落ちる結果になってしまいます。

また構築の際、それぞれのブロックデバイスは独自のものを使用する必要があります。このようなシステムの構築は複雑な作業であるため、ここでは説明していません。

31.3 Xen HA とプライベートブリッジ

お互いに通信すべきゲストシステムが存在する場合、もちろん通常のインターフェイスを介して通信してもかまいません。ですが、セキュリティ上の理由から、必要なゲストシステムのみを参加させたブリッジを作成して、そのブリッジ経由で通信を行うように設定しておくことをお勧めします。

ライブマイグレーションに対応すべき高可用性環境の場合、このようなプライベートブリッジはもう一方の Xen ホストにも接続されていなければなりません。これはプライベートブリッジ専用の物理イーサネットデバイスを用意して、専用のネットワークを構築することによって実現することができます。

もう一つの方法としてあげられるのが、VLAN インターフェイスの使用です。この場合、全てのトラフィックは通常のイーサネットインターフェイス経由で送信されます。ただし、VLAN のパケットにのみタグが付与され、通常のトラフィックとは区別されるようにすることができますので、これによって通常のトラフィックと混在させることができるようになります。

VLAN インターフェイスの設定方法について、詳しくは 8.1.1.4項「VLAN インターフェイスの使用」 をお読みください。

32 Xen: 準仮想化 (PV) ゲストから完全仮想化 (FV/HVM) ゲストへの変換

改訂履歴

2024-06-25

本章では、Xen の準仮想化ゲスト (仮想マシン) を Xen の完全仮想化マシンに変換する方法について説明しています。

手順 32.1: ゲスト側

ゲストを FV モードで起動できるようにするには、ゲスト内で下記の手順を実施する必要があります。

1. まずはゲストを変換する前に、全ての修正を適用してゲストを再起動してください。
2. FV の仮想マシンでは `-default` カーネルを使用します。このカーネルをインストールしていない場合は、PV モードで動作させている間に `kernel-default` パッケージをインストールします。
3. PV の仮想マシンでは `vda*` のようなデバイス名でディスクにアクセスしていますが、これらの名前を `hd*` のような名前に変換する必要があります。この変更は、それぞれ下記に示すファイルに対して行います:
   - `/etc/fstab`
   - `/boot/grub/menu.lst` (SLES 11 のみ)
   - `/boot/grub*/device.map`
   - `/etc/sysconfig/bootloader`
   - `/etc/default/grub` (SLES 12, 15 および openSUSE)



注記: UUID の使用について

`/etc/fstab` ファイル内では、UUID や論理ボリューム名での指定の使用をお勧めします。UUID を指定することにより、ネットワークストレージやマルチパスデバイス、仮想化などの様々なメリットを享受することができます。ディスクの UUID を表示させたい場合は、`blkid` コマンドをお使いください。

4. また、必要なモジュールを含む `initrd` の再生成にあたって発生するエラーを回避するため、`/dev/hda2` から `/dev/xvda2` へのシンボリックリンクを作成することもできます。シンボリックリンクの作成は、`ln` コマンドで行います:

```
ln -sf /dev/xvda2 /dev/hda2
ln -sf /dev/xvda1 /dev/hda1
.....
```

5. PV のマシンと FV のマシンではそれぞれ異なるディスクドライバモジュールやネットワークドライバモジュールを使用します。これらの FV モジュールは initrd 内に手作業で追加する必要があります。必要なモジュールはそれぞれ `xen-vbd` (ディスクデバイス用) および `xen-vnif` (ネットワークデバイス用) です。これらは完全仮想化 (FV) モードの VM ゲスト 向けの PV ドライバであり、それ以外の `ata_piix` , `ata_generic`, `libata` 等のモジュールは自動的に追加されます。



ヒント: initrd へのモジュールの追加について

- SLES 11 では、`/etc/sysconfig/kernel` ファイル内の `INITRD_MODULES` 内に必要なモジュールを指定します。たとえば下記のようになります:

  ```
  INITRD_MODULES="xen-vbd xen-vnif"
  ```

  あとは `dracut` を実行することで、必要なモジュールを含む新しい initrd を生成することができます。

- SLES 12, 15 および openSUSE の場合は、`/etc/dracut.conf.d/10-virt.conf` という名前のファイルを開くか新規に作成して、下記のような行を記述して `force_drivers` 以下にドライバを追加していきます (なお、二重引用符の後ろにスペースがあることに注意してください):

  ```
  force_drivers+=" xen-vbd xen-vnif"
  ```

  あとは `dracut -f --kver カーネルバージョン-default` のように入力して実行することで、必要なモジュールを含む新しい initrd を生成することができます。

  カーネルバージョンの検出方法について: `uname -r` コマンドを実行することで、お使いのシステムで現在動作中のカーネルのバージョンを調べることができます。

6. また、ゲストをシャットダウンする前に、`yast bootloader` を利用して `-default` カーネルに対する既定の起動パラメータを設定してください。

7. openSUSE Leap 11 の環境で、ゲスト側で X サーバを動作させている場合、X ドライバを変更する目的で `/etc/X11/xorg.conf` ファイルを変更する必要があります。`fbdev` という行を検索して、値を `cirrus` に変更してください。

```
Section "Device"
          Driver       "cirrus"
          ......
          EndSection
```



注記: openSUSE Leap 12/15 と Xorg について

openSUSE Leap 12/15 では、Xorg は必要なドライバを自動検出します。

8. ゲストをシャットダウンします。

手順 32.2: ホスト側

下記では、ホスト側で実施すべき手順を説明しています。

1. ゲストを FV モードで起動できるようにするため、VM の設定を FV 用に調整する必要があります。VM の設定編集は `virsh edit [ドメイン]` で行うのが簡単です。それぞれ下記の内容を編集してください:

   - OS セクション内のマシンの種類と `loader` の項目をそれぞれ編集し、`xenpv` を `xenfv` に変更します。変更後の OS セクションは下記のようになるはずです:

     ```
     <os>
               <type arch='x86_64' machine='xenfv'>hvm</type>
               <loader>/usr/lib/xen/boot/hvmloader</loader>
               <boot dev='hd'/>
     </os>
     ```

   - OS セクション内にある PV ゲスト固有の設定を削除します:

     - `<bootloader>pygrub</bootloader>`
     - `<kernel>/usr/lib/grub2/x86_64-xen/grub.xen</kernel>`
     - `<cmdline>xen-fbfront.video=4,1024,768</cmdline>`

   - デバイスセクション内に qemu エミュレータを追加します:

     ```
     <emulator>/usr/lib/xen/bin/qemu-system-i386</emulator>
     ```

- ディスクの設定を FV 書式になるよう、ターゲットデバイスとバスをそれぞれ変更します。具体的には `xen` ディスクバスを `ide` に、`vda` ターゲットデバイスを `hda` にそれぞれ変換します。変更後は下記のようになるはずです:

```
<target dev='hda' bus='ide'/>
```

- キーボードやマウスの接続に使用するバスを、`xen` から `ps2` に変更します。これに加えて、新しい USB タブレットデバイスを追加しておきます:

```
<input type='mouse' bus='ps2'/>
          <input type='keyboard' bus='ps2'/>
<input type='tablet' bus='usb'/>
```

- コンソールターゲットの種類を `xen` から `serial` に変更します:

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

- ビデオの設定を `xen` から `cirrus` に変更し、VRAM に 8 MB 程度を割り当てます:

```
<video>
          <model type='cirrus' vram='8192' heads='1' primary='yes'/>
</video>
```

- また、必要であれば VM の機能に `acpi` と `apic` を追加します:

```
<features>
          <acpi/>
          <apic/>
</features>
```

2. あとはゲストを起動する (`virsh` もしくは `virt-manager` を使用します) だけです。ゲストが kernel-default を利用して起動するようになったら (`uname -a` で確認できます) 、完全仮想化モードへの変換は完了となります。



注記: guestfs-tools

この処理を自動化したい場合やディスクイメージを直接編集したい場合は、guestfs-tools スイート (詳しくは 20.3項「guestfs ツール」をお読みください) をお使いください。ここにはディスクイメージを修正するための様々なツールが用意されています。

V QEMU を利用した仮想マシンの管理

33 QEMU の概要 304

34 KVM VM ホストサーバ の構築 305

35 ゲストのインストール 315

36 qemu-system-ARCH を利用した仮想マシンの実行 331

37 QEMU モニタを利用した仮想マシンの管理 360

33 QEMU の概要

改訂履歴

2023-06-06

QEMU は高速に動作するクロスプラットフォーム対応のオープンソース型マシンエミュレータです。数多くのハードウエアアーキテクチャに対応し、オペレーティングシステムを一切修正することなく、既存のシステム (VM ホストサーバ) 内にもう 1 つのオペレーティングシステム (VM ゲスト) を動作させることができます。また。QEMU はデバッグ目的でもしようすることができます。動作中の仮想マシンをいったん停止して状況を調査したり、状態の保存や復元を行ったりすることもできます。

QEMU は主に下記の部品から構成されています:

- プロセッサエミュレータ
- グラフィックカード, ネットワークカード, ハードディスク, マウスなどの擬似デバイス
- 対応するホストデバイスの擬似デバイスとして動作する汎用デバイス
- デバッガ
- エミュレータを操作するためのユーザインターフェイス

QEMU は一般的なマシンの擬似を提供することから、KVM や Xen の仮想化での中心的な役割を演じています。Xen での QEMU の使用はユーザ側からは隠蔽されていますが、KVM ではほとんどの QEMU の機能を透過的に提供しています。なお、VM ゲスト のアーキテクチャが VM ホストサーバ と同じであれば、QEMU は KVM アクセラレーションによってより高速に動作するようになります (なお、SUSE では、KVM アクセラレーション機能が有効化されている場合のみをサポート対象としています) 。

仮想化インフラストラクチャやプロセッサ固有の中枢機能を提供する以外にも、QEMU ではアーキテクチャ固有のユーザスペースプログラムを用意し、VM ゲスト の管理用に使用しています。なお、アーキテクチャによって別々のプログラムとなっていて、具体的には下記のような名前になっています:

- `qemu-system-i386`
- `qemu-system-s390x`
- `qemu-system-x86_64`
- `qemu-system-aarch64`

移行では、このコマンドは `qemu-system-アーキテクチャ` と呼んでいます。お使いの環境に合わせて、`qemu-system-x86_64` などに読み替えてください。

34 KVM VM ホストサーバ の構築

改訂履歴

2024-06-27

本章では、openSUSE Leap 15.7 を QEMU-KVM ベースの仮想マシンホストとして動作させるまでの手順と、その使用方法について説明しています。



ヒント: リソースについて

一般的に、仮想マシンのゲストシステムには、物理マシンにインストールする場合と同じだけのハードウエアリソースが必要となります。VM ホストサーバ システム内で多くのゲストシステムを稼働させる場合、それだけ多くの CPU コアやディスク、メモリやネットワークなどを必要とすることになります。

34.1 仮想化のための CPU 側サポート

KVM を動作させるには、お使いの CPU が仮想化機能に対応し、BIOS 側でも仮想化機能が有効化されていなければなりません。CPU の機能について調べるには、`/proc/cpuinfo` ファイル内に情報が書かれています。

34.2 必要なソフトウエア

KVM ホストに対しては、いくつかのパッケージをインストールする必要があります。必要な全てのパッケージをインストールするには、下記の手順を実施します:

1. まずは `yast2-vm` パッケージがインストールされていることを確認します。このパッケージは YaST の設定ツールで、仮想化のハイパーバイザのインストールを簡略化することができるものです。
2. [YaST] › [仮想化] › [ハイパーバイザとツールのインストール] を選択します。



図 34.1: KVM ハイパーバイザとツールのインストール

3. ［KVM サーバ］を選択します。なお、必要であれば［KVM ツール］も選択します。選択を行ったら［了解］を押します。

4. インストール処理が始まります。なお、処理の途中で YaST 側から、［ネットワークブリッジ］の自動作成を行うかどうかを尋ねられます。仮想化ゲストに対して専用のネットワークインターフェイスを割り当てるような場合は不要ですが、それ以外の場合はここでブリッジを作成して、ゲストマシンをネットワークに接続するのが一般的です。



図 34.2: ネットワークブリッジ

5. 必要なパッケージを全てインストールし、必要であれば新しいネットワークブリッジの作成を行ったら、お使いの CPU の種類に合わせて KVM のカーネルモジュールを読み込みます。具体的には `kvm_intel` もしくは `kvm_amd` のいずれかを読み込みます:

```
# modprobe kvm_intel
```

モジュールが正しく読み込めていることを確認します:

```
> lsmod | grep kvm
kvm_intel              64835  6
kvm                   411041  1 kvm_intel
```

これで KVM ホストの準備が整い、KVM の VM ゲストを開始することができるようになります。続きは 第36章「qemu-system-ARCH を利用した仮想マシンの実行」をお読みください。

34.3 KVM ホスト固有の機能

VM ホストサーバのハードウエアの機能を完全に生かし切る (準仮想化 を使用します) ことによって、KVM ベースの VM ゲストの性能を改善することができます。本章では、エミュレーションを介することなく、ゲスト側から物理ホストのハードウエアを直接アクセスするための技術について説明しています。



ヒント

本章内で示しているコマンド例では、`qemu-system-ARCH` コマンドのオプションについて知っていることを前提にして説明しています。詳しくは 第36章「qemu-system-ARCH を利用した仮想マシンの実行」をお読みください。

34.3.1 `virtio-scsi` を利用したホスト側ストレージの使用

`virtio-scsi` は KVM 向けの高度なストレージスタックです。これは従来 SCSI デバイスのパススルーで使用していた `virtio-blk` スタックの置き換えとして生まれた仕組みでもあります。`virtio-blk` と比較すると、下記の利点があります:

スケーラビリティの改善

KVM ゲストに設定できる PCI コントローラには制限が存在することから、接続できるデバイス数にも制限が生まれる結果になっています。`virtio-scsi` では、複数のストレージデバイスを単一のコントローラにまとめることによって、この制限を乗り越えることができるようになっています。`virtio-scsi` コントローラ内のデバイスは論理ユニット (LUN) として現れるようになっています。

標準的なコマンドセット

`virtio-blk` では `virtio-blk` のドライバと仮想マシンモニタの両方で共通する少数のコマンドセットのみを使用していたため、新しいコマンドを追加するにも、ドライバとモニタの両方を更新する必要がありました。

`virtio-scsi` ではコマンドを定義せずに伝送プロトコルを定義し、それらのコマンドを工業規格である SCSI 仕様に従って送信するようになっています。このようなアプローチにより、ファイバーチャネルや ATAPI, USB デバイスなどの他の技術とも共用がはかれるようになっています。

デバイスの名前

`virtio-blk` でのデバイスはゲスト内で `/dev/vdX` として現れるようになっていました。そのため物理システムとはデバイス名が異なることにより、移行時の障害になっていました。

`virtio-scsi` では物理システムと同じデバイス名になるようになっています。そのため、仮想マシンを容易に再配置できるようになっています。

SCSI デバイスのパススルー

ホスト内の LUN 全体を表す仮想ディスクの場合、ゲストから SCSI コマンドを直接 LUN に送信 (パススルー) したほうが良い場合があります。これは `virtio-blk` の場合、ゲスト側では SCSI コマンドではなく virtio-blk プロトコルを使用していることから実現できませんし、Windows ゲストの場合でも利用できません。`virtio-scsi` では SCSI そのものを使用しますので、何も問題なく対応できることになります。

34.3.1.1 virtio-scsi の使用方法

KVM では `virtio-scsi-pci` デバイスで SCSI のパススルー機能に対応しています:

```
# qemu-system-x86_64 [...] \
-device virtio-scsi-pci,id=scsi
```

34.3.2 vhost-net を利用したネットワーク処理の高速化

`vhost-net` モジュールは KVM の準仮想化ネットワークドライバを高速化するために使用するモジュールです。遅延とスループットの改善をそれぞれ提供します。`vhost-net` ドライバを使用するには、下記のようにしてコマンドラインで指定します:

```
# qemu-system-x86_64 [...] \
-netdev tap,id=guest0,vhost=on,script=no \
-net nic,model=virtio,netdev=guest0,macaddr=00:16:35:AF:94:4B
```

ここで、`guest0` はデバイスの識別用文字列です。

34.3.3 マルチキュー型 virtio-net を利用したネットワーク性能の強化

VM ゲスト 内での仮想 CPU 数を増やす場合、QEMU では マルチキュー と呼ばれる仕組みを利用して、ネットワーク性能を改善することができます。マルチキュー型の virtio-net では、複数の VM ゲスト の仮想 CPU が同時にパケットを送信することができるようになりますので、ネットワークの性能を大きく向上させることができます。なお、マルチキューへの対応は、VM ホストサーバ と VM ゲスト の両方で行う必要があります。



ヒント: 性能面での利点について

マルチキュー型の virtio-net ソリューションは、下記のような場合に有用です:

- ネットワークトラフィックのパケットが大きい場合。
- VM ゲスト 内で多数の接続を同時に処理する必要がある場合。これはゲストシステム間だけでなく、ゲストとホストの間やゲストと外部システムの間でも有用です。
- VM ゲスト 内での有効なキュー数と仮想 CPU の数が一致している場合。



注記

マルチキュー型の virtio-net は全体のネットワーク性能を強化する仕組みであるため、仮想 CPU の使用率も上がることになります。

手順 34.1: マルチキュー型 VIRTIO-NET の有効化方法

下記の手順では `qemu-system-ARCH` を利用してマルチキュー機能を有効化する場合の、主な流れを説明しています。なお、VM ホストサーバ 側にはマルチキュー機能に対応した tap ネットワークデバイス (カーネルバージョン 3.8 以降で対応しています) が既に設定されているものとします。

1. tap デバイスに対してマルチキュー機能を有効化するには、`qemu-system-ARCH` に下記を追加します:

   ```
   -netdev tap,vhost=on,queues=2*N
   ```

   ここで、`N` にはキュー対の数を指定します。

2. さらに `qemu-system-ARCH` 側でマルチキューを有効化し、virtio-net-pci デバイスに対する MSI-X (Message Signaled Interrupt (メッセージシグナル型割り込み)) ベクトルを指定します:

   ```
   -device virtio-net-pci,mq=on,vectors=2*N+2
   ```

   ここで、MSI-X ベクトルの数 (vectors=) は下記のようにして計算します: まず TX (送信) キュー用に N 個、RX (受信) 用に N 個、設定用に 1 個、そして発生しうる VQ (Vector Quantization (ベクトル量子化)) 制御用に 1 個を用意します。つまり、合計で 2*N+2 になります。

3. VM ゲスト 側でネットワークインターフェイスに対してマルチキューを有効化します (下記はデバイスが `eth0` である場合の例です):

   ```
   > sudo ethtool -L eth0 combined 2*N
   ```

生成された `qemu-system-ARCH` のコマンドラインは、下記のようになるはずです:

```
qemu-system-x86_64 [...] -netdev tap,id=guest0,queues=8,vhost=on \
-device virtio-net-pci,netdev=guest0,mq=on,vectors=10
```

ここで、ネットワークデバイスの `id` (`guest0`) は、両方に同じ値を指定する必要があることに注意してください。

VM ゲスト 内では、`root` 権限で下記のようなコマンドを入力して実行します:

```
> sudo ethtool -L eth0 combined 8
```

これでゲストシステムのネットワークインターフェイスは、`qemu-system-ARCH` ハイパーバイザが提供するマルチキュー機能に対応するようになっています。

34.3.4 VFIO: デバイスに対する安全な直接アクセス

VM ゲスト に対する PCI デバイスの直接割り当て (PCI パススルー) を使用することで、性能を重視する環境ではエミュレーションに必要な処理を回避することができます。VFIO は従来の KVM PCI パススルー デバイス割り当てを置き換えるよう作成された仕組みで、この機能を利用するには 重要: VFIO および SR-IOV の要件について で説明している要件を満たす必要があります。

VFIO を利用して PCI デバイスを VM ゲスト に割り当てることができるようにするには、まず所属する IOMMU グループが何であるのかを調べる必要があります。IOMMU (Input/Output Memory Management Unit; メインメモリへの直接メモリアクセスに対応した I/O バス) API にはグループという概念があります。グループはデバイスの集合で、システム内での境界線として使用しているものです。そのため、グループは VFIO が使用する所有権の単位であると言えます。

手順 34.2: VFIO を介した VM ゲスト への PCI デバイスの割り当て

1. まずはゲストに割り当てるホスト側の PCI デバイスを識別します。

```
> sudo lspci -nn
[...]
00:10.0 Ethernet controller [0200]: Intel Corporation 82576 \
Virtual Function [8086:10ca] (rev 01)
[...]
```

   対象のデバイスのデバイス ID (この例では `00:10.0`) と製造元 ID (`8086:10ca`) をそれぞれメモしておきます。

2. このデバイスの IOMMU グループを判別します:

```
> sudo readlink /sys/bus/pci/devices/0000\:00\:10.0/iommu_group
../../../kernel/iommu_groups/20
```

   このデバイスの IOMMU グループは `20` であることがわかりました。あとは同じ IOMMU グループに属しているデバイスを確認します:

```
> sudo ls -l /sys/bus/pci/devices/0000\:01\:10.0/iommu_group/devices/
[...] 0000:00:1e.0 -> ../../../../devices/pci0000:00/0000:00:1e.0
[...] 0000:01:10.0 -> ../../../../devices/pci0000:00/0000:00:1e.0/0000:01:10.0
[...] 0000:01:10.1 -> ../../../../devices/pci0000:00/0000:00:1e.0/0000:01:10.1
```

3. デバイスドライバからデバイスを取り外します:

```
> sudo echo "0000:01:10.0" > /sys/bus/pci/devices/0000\:01\:10.0/driver/unbind
```

4. 上記で取得した製造元 ID を指定して、デバイスを vfio-pci ドライバに接続します:

```
> sudo echo "8086 153a" > /sys/bus/pci/drivers/vfio-pci/new_id
```

   上記を実行すると、`/dev/vfio/IOMMU_グループ` という新しいデバイスが作成されます。上記の例の場合は `/dev/vfio/20` になります。

5. 新しく作成したデバイスの所有者を変更します:

```
> sudo chown qemu.qemu /dev/vfio/デバイス名
```

6. あとは割り当てた PCI デバイスを指定して VM ゲスト を起動します。

```
> sudo qemu-system-ARCH [...] -device
     vfio-pci,host=00:10.0,id=ID
```

重要: ホットプラグ (活性接続) への非対応について

openSUSE Leap 15.7 では、VFIO を介した PCI デバイスのホットプラグによる VM ゲストへの接続はサポートしていません。

VFIO ドライバに関するさらに詳しい情報については、`/usr/src/linux/Documentation/vfio.txt` (英語) をお読みください (`kernel-source` パッケージをインストールする必要があります) 。

34.3.5 VirtFS: ホストとゲストの間でのディレクトリ共有

VM ゲスト は通常、個別の作業領域で動作します。メモリ範囲や CPU を独自に割り当てるほか、ファイルシステムも別々になります。VM ホストサーバ 側のファイルシステムとの間で共有を行うことで、相互のデータ交換をより柔軟に行うことができるようになります。CIFS や NFS などのネットワーク型のファイルシステムでもディレクトリの共有を行うことができますが、これらは仮想化用に設計されたものではありませんので、十分な性能を提供することができないほか、機能面の問題に直面する可能性もあります。

KVM では新しく最適化された VirtFS (「ファイルシステムのパススルー」と呼ぶこともあります) を提供しています。VirtFS は準仮想化型のファイルシステムドライバで、ゲスト側でのファイルシステム操作をブロックデバイスの操作に変換する処理を行うことなく、そのままホスト側のファイルシステム操作に変換することができます。

VirtFS の機能は下記のような用途で使用することができます:

- 複数のゲストから共通のディレクトリにアクセスする必要がある場合や、ゲスト間で同じファイルシステムにアクセスする必要がある場合。
- ゲスト側の RAM ディスクで、起動時にアクセスするルートファイルシステムの代替として使用したい場合。
- クラウド環境で、ホスト側のファイルシステムを複数の顧客に提供するようなストレージサービスを構成するような場合。

34.3.5.1 実装

QEMU では、VirtFS の実装を下記の 2 種類のデバイスとして提供しています:

- `virtio-9p-pci` : プロトコルのメッセージとデータを、ホストとゲストの間で伝送するデバイス。
- `fsdev` : ファイルシステムの種類やセキュリティモデルなど、ファイルシステムの属性情報を提供するためのデバイス。

例 34.1: VIRTFS を利用したホスト側ファイルシステムの公開

```
> sudo qemu-system-x86_64 [...] \
-fsdev local,id=exp1❶,path=/tmp/❷,security_model=mapped❸ \
-device virtio-9p-pci,fsdev=exp1❹,mount_tag=v_tmp❺
```

❶ 公開するファイルシステムの識別子を指定します。

❷ 公開するファイルシステムのホスト内でのパスを指定します。

❸ 使用するセキュリティモデルを指定します。`mapped` を指定すると、ゲスト側からのファイルシステムモードとパーミッションを維持し、ホスト側とは区別して管理するようにします。`none` は「パススルー」型のセキュリティモデルで、ゲスト側のファイルアクセスでのパーミッションをそのままホスト側にも反映させる設定です。

❹ `-fsdev id=` で指定していたファイルシステムの識別子を指定します。

❺ ゲスト側でマウントを行う際、タグとして使用する名前を指定します。

上記のようにファイルシステムを公開した場合、ゲスト側では下記のようにしてマウントすることができます:

```
> sudo mount -t 9p -o trans=virtio v_tmp /mnt
```

ここで、`v_tmp` は `-device mount_tag=` で指定したタグ名を、`/mnt` は公開されたファイルシステムをマウントする先をそれぞれ指定します。

34.3.6 KSM: ゲスト間でのメモリページ共有

Kernel Same Page Merging (KSM) は Linux カーネルの機能で、複数のプロセス内にある同一のメモリページを、1 つのメモリ領域にまとめる機能のことを指します。KVM ゲストは Linux 内のプロセスとして存在することになりますので、KSM を使用することでハイパーバイザがメモリをより効率的に使用し、オーバーコミット機能 (搭載されているメモリよりも多くのメモリを提供する機能) を提供することにもなります。そのため、メモリの限られたホスト内で複数の仮想マシンを動作させる必要がある場合、KSM を利用することで解決できる場合があることになります。

KSM では、`/sys/kernel/mm/ksm` ディレクトリ内に、その状態に関する情報を含むファイルを提供します:

```
> ls -1 /sys/kernel/mm/ksm
full_scans
merge_across_nodes
pages_shared
pages_sharing
pages_to_scan
pages_unshared
pages_volatile
```

```
run
sleep_millisecs
```

`/sys/kernel/mm/ksm/*` ファイルのそれぞれの意味について、詳しくは `/usr/src/linux/Documentation/vm/ksm.txt` (英語) をお読みください (`kernel-source` パッケージ内に含まれています) 。

KSM を使用するには、下記の手順を実施します:

1. openSUSE Leap では、KSM のサポートがカーネル内に含まれていますが、既定では無効化されています。有効化するには、下記のコマンドを実行します:

   ```
   # echo 1 > /sys/kernel/mm/ksm/run
   ```

2. あとは必要な VM ゲスト を KVM 内で動作させます。機能が正しく動作しているかどうかを調べるには、`pages_sharing` と `pages_shared` のファイルの内容を調べてください:

   ```
   > while [ 1 ]; do cat /sys/kernel/mm/ksm/pages_shared; sleep 1; done
   13522
   13523
   13519
   13518
   13520
   13520
   13528
   ```

35 ゲストのインストール

改訂履歴

2024-06-27

`virt-manager` や `virt-install` など、`libvirt` ベースのツールを利用することで、仮想マシンの構築から管理までを便利に行うことができます。実態としては、これらは `qemu-system-ARCH` コマンドのラッパーとして動作していますが、`libvirt` ベースのツールを使用せずに `qemu-system-ARCH` を直接使用することもできます。



警告: `qemu-system-ARCH` と libvirt の関係性について

`qemu-system-ARCH` で作成した 仮想マシン は、`libvirt` ベースのツールから参照することはできません。

35.1 `qemu-system-ARCH` を利用した基本的なインストール

下記の例では、SUSE Linux Enterprise Server 11 をインストールした仮想マシンを作成します。各コマンドに対する詳細に名説明については、それぞれのマニュアルページをお読みください。

仮想環境内で使用するシステムのイメージを作成していない場合は、まずインストールメディアでイメージを作成します。この場合、ハードディスクイメージを新規に作成して、インストールメディアのイメージを取得するか、インストールメディアそれ自身を用意します。

まずは `qemu-img` でハードディスクを作成します。

```
> qemu-img create❶ -f raw❷ /images/sles/hda❸ 8G❹
```

❶ `create` を指定することで、`qemu-img` に対して新しいイメージを作成するよう指示しています。

❷ `-f` パラメータでディスク形式を指定しています。

❸ イメージファイルのフルパスを指定しています。

❹ イメージのサイズ (この場合は 8 GB) を指定しています。イメージは スパースイメージファイル として作成され、ディスクにデータが書き込まれていくことで実際のサイズが増えていく形態になります。ここで指定したサイズは、イメージファイルの最大サイズを指定していることになります。

少なくとも 1 台のハードディスクを作成したら、あとは `qemu-system-ARCH` を実行して仮想マシンを起動し、インストールシステムを開始します:

```
# qemu-system-x86_64 -name "sles"❶ -machine accel=kvm -M pc❷ -m 768❸ \
```

```
-smp 2❹ -boot d❺ \
-drive file=/images/sles/hda,if=virtio,index=0,media=disk,format=raw❻ \
-drive file=/isos/SLE-15-SP7-Online-ARCH-GM-media1.iso,index=1,media=cdrom❼ \
-net nic,model=virtio,macaddr=52:54:00:05:11:11❽ -net user \
-vga cirrus❾ -balloon virtio❿
```

❶ ウインドウのタイトル部や VNC サーバに接続した際に表示される仮想マシンの名前を指定しています。この名前は他の仮想マシンと重複していてはなりません。

❷ マシンの種類を指定します。設定可能な値の一覧を表示するには、`qemu-system-ARCH -M ?` と入力して実行してください。なお `pc` を指定すると、既定値である［Standard PC］(標準 PC) を指定したものとみなされます。

❸ 仮想マシンに対して割り当てる最大メモリ量を指定しています。

❹ 2 台のプロセッサが搭載された SMP システムであることを指定しています。

❺ 起動の順序を指定しています。設定可能な値は `a` , `b` (フロッピィディスク 1 台目および 2 台目), `c` (1 台目のハードディスク), `d` (1 台目の CD-ROM), `n` から `p` まで (ネットワークアダプタの 1 枚目から 3 枚目によるネットワーク起動) のいずれかです。既定値は `c` です。

❻ 1 台目 (`index=0`) のハードディスクを定義しています。ここでは、準仮想化 (`if=virtio`) ドライブとして提供し、`raw` 形式を使用する設定が書かれています。

❼ 2 台目 (`index=1`) のドライブは CD-ROM として動作するイメージドライブであることを指定しています。

❽ 準仮想化 (`model=virtio`) ネットワークアダプタを、MAC アドレス `52:54:00:05:11:11` で定義しています。なお、MAC アドレスの重複が発生しないように設定してください。重複が発生すると、通信ができなくなってしまいます。

❾ グラフィックカードを指定しています。`none` を指定すると、グラフィックカードが無効化されます。

❿ 準仮想化型のバルーンデバイスを定義し、メモリ量を動的に変更できるように設定しています (ただし `-m` で指定したサイズは超えないものとします) 。

ゲスト側のオペレーティングシステムのインストールが完了したら、あとは CD-ROM ドライブ無しで仮想マシンを起動することができます:

```
# qemu-system-x86_64 -name "sles" -machine type=pc,accel=kvm -m 768 \
-smp 2 -boot c \
-drive file=/images/sles/hda,if=virtio,index=0,media=disk,format=raw \
-net nic,model=virtio,macaddr=52:54:00:05:11:11 \
-vga cirrus -balloon virtio
```

35.2 qemu-img を利用したディスクイメージの管理

上記の章 (35.1項「qemu-system-ARCH を利用した基本的なインストール」) では qemu-img コマンドを利用してハードディスクのイメージを作成してきましたが、このコマンドは一般的なディスク操作にも使用することができます。本章では、ディスクイメージをより柔軟に管理するための qemu-img のサブコマンドについて紹介しています。

35.2.1 qemu-img の実行に関する一般的な情報

qemu-img は zypper と同様に、サブコマンドを指定してさまざまな処理を行います。それぞれのサブコマンドにはそれぞれのオプションが用意されています。オプションによっては一般的なもののほか、サブコマンド特有のものも存在しています。オプションの一覧について、詳しくはマニュアルページ (man 1 qemu-img) をお読みください。また、qemu-img は、下記のような書式で実行します:

```
> qemu-img サブコマンド [オプション]
```

また、下記のようなサブコマンドに対応しています:

`create`
: ファイルシステム内に新しいディスクイメージを作成します。

`check`
: 既存のディスクイメージ内にエラーがないかどうかをチェックします。

`compare`
: 2 つのイメージファイル内に同じ内容が含まれているかどうかを確認します。

`map`
: イメージファイル名とそのファイルチェインのメタデータをダンプ出力します。

`amend`
: 指定したイメージファイルに対して、イメージ形式固有のオプションを修正します。

`convert`
: 既存のディスクイメージを異なる形式に変換して新しいディスクイメージを作成します。

`info`
: ディスクイメージに関する情報を表示します。

`snapshot`
: 既存のディスクイメージに対するスナップショットを管理します。

`commit`

既存のディスクイメージに対する変更を適用します。

`rebase`

既存のイメージを利用して新しいベースイメージを作成します。

`resize`

既存のディスクイメージのサイズを拡張／縮小します。

35.2.2 ディスクイメージの作成／変換／検査

本章では、ディスクイメージの作成方法や状態の確認方法、ディスクイメージ形式の変換方法、そしてディスクイメージの詳細情報の取得方法について説明しています。

35.2.2.1 qemu-img create

`qemu-img create` は VM ゲスト のオペレーティングシステム向けに新しいディスクイメージを作成するコマンドです。このコマンドは下記のような書式で実行します:

```
> qemu-img create -f fmt❶ -o options❷ fname❸ size❹
```

❶ 作成するイメージの形式を指定します。サポートされている形式は、`raw` , `qcow2` の 2 種類です。

❷ イメージ形式によっては、コマンドラインに追加のオプションを指定することができるものもあります。この追加のオプションは `-o` オプションで指定します。`raw` イメージ形式の場合は `size` オプションにのみ対応しています。そのため、コマンドの末尾でサイズを指定する代わりに、`-o size=8G` のように指定してサイズを設定することもできます。

❸ 作成すべきディスクイメージファイルのフルパスを指定します。

❹ 作成するディスクイメージのサイズを指定します (`-o size=<イメージサイズ>` オプションを指定しなかった場合) 。サイズ指定では、それぞれ下記の単位接尾辞を使用することができます: `K` (キロバイト), `M` (メガバイト), `G` (ギガバイト), `T` (テラバイト)

たとえば新しいディスクイメージ `sles.raw` を `/images` ディレクトリ内に作成し、最大サイズを 4 GB として設定したい場合は、下記のコマンドを入力して実行します:

```
> qemu-img create -f raw -o size=4G /images/sles.raw
Formatting '/images/sles.raw', fmt=raw size=4294967296
```

```
> ls -l /images/sles.raw
-rw-r--r-- 1 tux users 4294967296 Nov 15 15:56 /images/sles.raw

> qemu-img info /images/sles.raw
image: /images/sles11.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 0
```

上記の通り、新しく作成したイメージの 仮想的な サイズは 4 GB ですが、実際のサイズは 0 バイトになっています。これは、イメージに対してまだ何もデータを書き込んでいないためです。



ヒント: btrfs ファイルシステム内での VM ゲスト イメージについて

btrfs ファイルシステム内にディスクイメージを作成する必要がある場合、btrfs に搭載されたコピーオンライト機能によって、性能劣化が発生する場合があります。この場合に備えて、`nocow=on` オプションを指定してイメージを作成することができます:

```
> qemu-img create -o nocow=on test.img 8G
```

なお、コピーオンライト機能を使用したい (たとえばスナップショットの作成や仮想マシン間を跨いだ共有など) 場合は、`nocow` オプションを指定せずに実行してください。

35.2.2.2 qemu-img convert

`qemu-img convert` はディスクイメージを他の形式に変換するためのコマンドです。QEMU でサポートされるイメージ形式の一覧を表示したい場合は、`qemu-img -h` と入力して実行し、出力の末尾のほうをご覧ください。このコマンドは下記のような書式になっています:

```
> qemu-img convert -c❶ -f fmt❷ -O out_fmt❸ -o options❹ fname❺ out_fname❻
```

❶ 作成するディスクイメージを圧縮するための指定です。`qcow` および `qcow2` 形式のみで指定することができます。

❷ 元となるディスクイメージの形式を指定します。通常は自動検出されるため、省略してもかまいません。

❸ 作成するディスクイメージの形式を指定します。

❹ 作成するディスクイメージ形式に対する追加のオプションを指定します。`-o ?` とオプションを入力して実行すると、指定したディスクイメージ形式で利用できるオプションの一覧を表示することができます。

❺ 変換元となるディスクイメージのパスを指定します。

❻ 変換先のディスクイメージのパスを指定します。

```
> qemu-img convert -O vmdk /images/sles.raw \
/images/sles.vmdk

> ls -l /images/
-rw-r--r-- 1 tux users 4294967296 16. lis 10.50 sles.raw
-rw-r--r-- 1 tux users 2574450688 16. lis 14.18 sles.vmdk
```

選択したイメージ形式で利用可能なオプションの一覧を表示するには、下記のようなコマンドを入力して実行します (下記の `vmdk` を使用したい形式に置き換えて実行してください):

```
> qemu-img convert -O vmdk /images/sles.raw \
/images/sles.vmdk -o ?
Supported options:
size             Virtual disk size
backing_file     File name of a base image
compat6          VMDK version 6 image
subformat        VMDK flat extent format, can be one of {monolithicSparse \
    (default) | monolithicFlat | twoGbMaxExtentSparse | twoGbMaxExtentFlat}
scsi             SCSI image
```

35.2.2.3 qemu-img check

`qemu-img check` コマンドは、既存のディスクイメージ内にエラーが無いかどうかを調査します。なお、ディスクイメージ形式によっては、この機能に対応していないものもあります。このコマンドは下記のような書式で実行します:

```
> qemu-img check -f fmt❶ fname❷
```

❶ 元となるディスクイメージの形式を指定します。通常は自動検出されるため、省略してもかまいません。

❷ 確認を行うディスクイメージのパスを指定します。

何もエラーが検出されない場合、コマンドは何も出力を行いません。それ以外の場合、それぞれのエラーと発生したエラーの数が表示されます。

```
> qemu-img check -f qcow2 /images/sles.qcow2
ERROR: invalid cluster offset=0x2af0000
[...]
ERROR: invalid cluster offset=0x34ab0000
378 errors were found on the image.
```

35.2.2.4 既存のディスクイメージのサイズ拡張

新しいイメージを作成する場合、イメージを作成する前に最大サイズを指定しなければなりません (詳しくは 35.2.2.1項「qemu-img create」をお読みください) 。ですが、VM ゲスト をインストールしてしばらく使用していると、最初に指定したサイズでは容量が不足してしまうことがあります。このような場合は、容量を増やして対応することができます。

既存のディスクイメージを 2 GB だけ拡張したい場合は、たとえば下記のように入力して実行します:

```
> qemu-img resize /images/sles.raw +2GB
```



注記

ディスクイメージサイズの変更を行うには、ディスクイメージの形式が `raw` , `qcow2` のいずれかでなければなりません。それ以外の形式になっているイメージのサイズを変更したい場合は、`qemu-img convert` でサイズ変更可能な形式に変換してから実施してください。

これでイメージファイルの末尾には 2 GB の空き領域が現れることになります。あとは既存のパーティションをサイズ変更するか、もしくは新しくパーティションを作成することで、その領域を使用できるようになります。

35.2.2.5 qcow2 ファイル形式向けの高度なオプション

qcow2 は QEMU で使用されるメインのディスクイメージ形式です。サイズを必要に応じて拡張する機能が用意されているほか、仮想マシン側で実際に必要とされているサイズのみを割り当てる機能があります。

qcow2 形式のファイルは固定のサイズを 1 つの単位としてまとめられています。これらの単位は クラスタ と呼ばれます。ゲスト側からの見た目での仮想ディスクは、このクラスタのサイズごとに区切られているものとして提供されます。QEMU の既定では、クラスタのサイズは 64 kB に設定されていますが、新しいイメージを作成する際に異なるサイズを設定することもできます:

```
> qemu-img create -f qcow2 -o cluster_size=128K virt_disk.qcow2 4G
```

qcow2 形式のイメージには、L1, L2 テーブルと呼ばれる、2 階層のテーブルセットが含まれています。L1 テーブルはディスクイメージごとに 1 つしか存在しませんが、L2 テーブルはイメージファイルの大きさに合わせて多数のものが存在しています。

仮想ディスクへの読み込みや書き込みが発生すると、QEMU は対応する L2 テーブルを読み込んで、必要なデータの場所を検出します。それぞれの I/O 操作に対応したテーブルの読み込み処理にはシステムのリソース消費が伴いますので、QEMU ではディスクアクセスの速度を向上させるため、メモリ内に L2 テーブルのキャッシュを保持するようになっています。

35.2.2.5.1 適切なキャッシュサイズの選択

キャッシュサイズは割り当て済みの領域の量にあわせて決まります。L2 キャッシュは下記の仮想ディスクサイズの計算式が出発点となります:

```
ディスクサイズ = L2_キャッシュのサイズ * クラスタサイズ / 8
```

既定のクラスタサイズである 64 kB であれば、上記の式は下記のようになります:

```
ディスクサイズ = L2_キャッシュのサイズ * 8192
```

このことから、既定のクラスタサイズであるとすると、ディスク領域をマッピングするためのキャッシュは、下記のようになります:

```
L2_キャッシュのサイズ = ディスクサイズ_(GB) * 131072
```

QEMU では、既定で L2 テーブルのキャッシュが 1 MB (1048576 バイト) に設定されています。上記の数式に当てはめると、1 MB のキャッシュはディスクサイズで 8 GB (1048576 / 131072) 分になります。つまり、作成する仮想ディスクのサイズが 8 GB までであれば、L2 テーブルのキャッシュサイズは十分であることになります。それより大きいディスクの場合は、L2 テーブルのキャッシュサイズを増やすことによって、ディスクの速度を上げることができます。

35.2.2.5.2 キャッシュサイズの設定

キャッシュサイズの指定を行うには、QEMU のコマンドラインに対して `-drive` オプションを指定します。それ以外の方法としては、QMP との通信を行っている場合、`blockdev-add` コマンドを使用することもできます。QMP に関する詳細は、37.11項「QMP - QEMU マシンプロトコル」をお読みください。

それぞれ下記のオプションを指定することで、仮想化ゲストのキャッシュサイズを設定することができます:

l2-cache-size
: L2 テーブルキャッシュの最大サイズを指定します。

refcount-cache-size
: refcount ブロックキャッシュの最大サイズを指定します。refcount に関する詳細については、https://raw.githubusercontent.com/qemu/qemu/master/docs/qcow2-cache.txt をお読みください。

cache-size
: 両方のキャッシュサイズの合計最大サイズを指定します。

上記のオプションを指定する場合は、下記に注意してください:

- L2 および refcount ブロックのキャッシュサイズは、クラスタサイズの倍数である必要があります。
- いずれかのオプションのみを指定した場合、QEMU は指定しなかったほうのオプションを自動調整します。具体的には、L2 キャッシュが refcount キャッシュよりも 4 倍大きくなるようにします。

refcount キャッシュは L2 キャッシュほど頻繁に使用されるものではありませんので、下記のように比較的小さな値にしておくことができます:

```
# qemu-system-ARCH [...] \
 -drive file=disk_image.qcow2,l2-cache-size=4194304,refcount-cache-size=262144
```

35.2.2.5.3 メモリ使用率の削減

キャッシュサイズを大きくすると、それだけメモリの使用量も増えてしまいます。それぞれの qcow2 ファイルには個別の L2 キャッシュが存在する仕組みであるため、多数の巨大なディスクイメージを使用していると、かなりの量のメモリを使用する結果になってしまいます。メモリの消費は、ゲストの構築作業内でバッキングファイル (35.2.4項「ディスクイメージの効率的な使用」) を使用したり、スナップショット (see 35.2.3項「qemu-img を利用した仮想マシンのスナップショット管理」) を作成したりすることで、より大きくなってしまいます。

このような理由から、QEMU では `cache-clean-interval` という設定が提供されています。これは全てのキャッシュ項目に対する設定で、指定した秒数以上アクセスのない項目があった場合、それらをメモリから削除するための設定です。

下記の例では、10 分間アクセスの無かった全てのキャッシュ項目を削除することになります:

```
# qemu-system-ARCH [...] -drive file=hd.qcow2,cache-clean-interval=600
```

このオプションを指定しない場合、既定値である 0 が適用され、この機能が無効化されます。

35.2.3 qemu-img を利用した仮想マシンのスナップショット管理

仮想マシン のスナップショットは VM ゲスト の動作中に採取することのできるスナップショットです。スナップショットにはプロセッサ (CPU) やメモリ (RAM) のほか、デバイスや全ての書き込み可能なディスクに関する状態が含まれます。

スナップショット機能は、お使いの仮想マシンで特定の状態を保存したい場合に有用です。たとえば仮想化サーバ内でネットワークサービスを設定し、いつでもその時点に戻すことができて欲しいような場合などがあります。それ以外にも、何らかの実験的な作業を行って不安定になってしまう前に、仮想マシ

ンをシャットダウンしてからバックアップを作成するようなことを行うこともできます。本章では後者について説明しています。前者については 第37章「QEMU モニタを利用した仮想マシンの管理」で説明しています。

スナップショットを使用するには、お使いの VM ゲスト 内に少なくとも 1 つ以上の書き込み可能な `qcow2` 形式のディスクイメージが存在しなければなりません。このようなデバイスは通常、1 台目の仮想ハードディスクであるはずです。

仮想マシン のスナップショットは、対話型の QEMU モニタ内で `savevm` コマンドを実行することで作成することができます。スナップショットを分かりやすくするために、タグ を指定しておくこともできます。QEMU モニタに関する詳細については、第37章「QEMU モニタを利用した仮想マシンの管理」をお読みください。

`qcow2` ディスクイメージ内にスナップショットを保存したあとは、`qemu-img snapshot` コマンドで内容を調査することができます。



警告: VM ゲスト のシャットダウンについて

仮想マシンが動作している場合、`qemu-img snapshot` コマンドで仮想マシンのスナップショットを作成したり、削除したりしてはなりません。動作中にこれらを実施してしまうと、仮想マシンの状態を持つディスクイメージが破壊されてしまう場合があります。

35.2.3.1 既存のスナップショット一覧の表示

`qemu-img snapshot -l` ディスクイメージ のように入力して実行すると、ディスクイメージ で指定したディスクイメージ内に保存されている全てのスナップショットを表示することができます。この一覧表示は、VM ゲスト の動作中でも表示することができます。

```
> qemu-img snapshot -l /images/sles.qcow2
Snapshot list:
ID❶        TAG❷                VM SIZE❸        DATE❹          VM CLOCK❺
1         booting                4.4M 2013-11-22 10:51:10   00:00:20.476
2         booted                 184M 2013-11-22 10:53:03   00:02:05.394
3         logged_in              273M 2013-11-22 11:00:25   00:04:34.843
4         ff_and_term_running    372M 2013-11-22 11:12:27   00:08:44.965
```

❶ スナップショットの識別番号です。通常は自動的に割り当てられます。

❷ スナップショットの説明 (タグ) です。識別用に用意されているものです。

❸ スナップショットが占有するディスク領域です。なお、動作中のアプリケーションがメモリを多く使用していればいるほど、スナップショットのサイズが大きくなります。

❹ スナップショットを作成した日時を表しています。

❺ 仮想マシンのクロック状態を表しています。

35.2.3.2 電源の入っていない仮想マシンに対するスナップショットの作成

仮想マシンの電源が切れている状態で、現在の状態に対するスナップショットを作成するには、`qemu-img snapshot -c` スナップショットのタイトル ディスクイメージ のように入力して実行します。

```
> qemu-img snapshot -c backup_snapshot /images/sles.qcow2
```

```
> qemu-img snapshot -l /images/sles.qcow2
Snapshot list:
ID        TAG                 VM SIZE                DATE       VM CLOCK
1         booting                4.4M 2013-11-22 10:51:10   00:00:20.476
2         booted                 184M 2013-11-22 10:53:03   00:02:05.394
3         logged_in              273M 2013-11-22 11:00:25   00:04:34.843
4         ff_and_term_running    372M 2013-11-22 11:12:27   00:08:44.965
5         backup_snapshot           0 2013-11-22 14:14:00   00:00:00.000
```

VM ゲスト 側で何らかの問題が発生し、保存されているスナップショットに戻す必要が生じた場合 (たとえば ID 5 に戻したい場合) 、VM ゲスト の電源を落としてから下記のようなコマンドを実行します:

```
> qemu-img snapshot -a 5 /images/sles.qcow2
```

上記を実行したあとは、通常通り `qemu-system-ARCH` で仮想マシンを起動することで、スナップショット番号 5 の状態から起動を行うことができます。



注記

なお、`qemu-img snapshot -c` のコマンドは QEMU モニタ (詳しくは 第37章「QEMU モニタを利用した仮想マシンの管理」をお読みください) の `savevm` コマンドとは関係がありません。たとえば QEMU モニタで `savevm` を実行して作成したスナップショットを、`qemu-img snapshot -a` で適用するようなことはできません。

35.2.3.3 スナップショットの削除

`qemu-img snapshot -d` スナップショット_ID ディスクイメージ のように入力して実行することで、仮想マシンに対する古いスナップショットや不要なスナップショットを削除することができます。これによりスナップショットが占有していた `qcow2` ディスクイメージ内の領域が解放されることになります:

```
> qemu-img snapshot -d 2 /images/sles.qcow2
```

35.2.4 ディスクイメージの効率的な使用

まずは下記のような状況を想像してみてください。あなたはサーバの管理者で、複数の仮想化オペレーティングシステムを動作させて管理を行っています。これらのシステムのうち、1 つのグループは特定のディストリビューションをベースにしていますが、その他のグループは異なる (おそらく Unix 以外の) プラットフォームを使用しています。さらに複雑なことに、それぞれの仮想ゲストシステムは同じディストリビューションをベースにしていますが、使用している部署や部門が異なるため、構成も異なっているような状態です。ファイルサーバについては Web サーバとは構成やサービスなどが異なっていますが、いずれも openSUSE をベースにしているものとします。

このような場合、「ベース」となるディスクイメージを作成して対応するようなことができます。このベースイメージは、仮想マシンの雛形として使用することができますので、何度もオペレーティングシステムをインストールしたりする手間を省くことができます。

35.2.4.1 ベースイメージと派生イメージ

まずは通常の手順でディスクイメージを作成し、目的のシステムをインストールしてください。詳しくは 35.1項「`qemu-system-ARCH` を利用した基本的なインストール」と 35.2.2項「ディスクイメージの作成／変換／検査」をお読みください。あとはそのイメージをベースイメージとして、新しいイメージファイルを作成するだけです。これで 派生 イメージを作成することができたことになります。なお、ベースイメージを利用して起動してはなりません。必ず派生イメージのほうを起動してください。また、1 つのベースイメージから複数の派生イメージを作成することもできます。そのため、ベースイメージのほうを変更してしまうと、派生関係が壊れてしまうことになります。派生イメージを利用して起動を行うと、QEMU は変更点を派生イメージのほうにのみ書き込むようになります。それに対して、ベースイメージは読み込み専用となります。

ベースイメージを作成する場合は、オペレーティングシステムを新規にインストールし (必要であれば登録作業などを行い) 、修正 (パッチ) などはインストールせず、アプリケーションについてもインストールや削除を行わないようにしておくことをお勧めします。必要であれば、そのベースイメージを元に、最新の修正を適用したもう 1 つのベースイメージを作成してもかまいません。

35.2.4.2 派生イメージの作成



注記

ベースイメージには `raw` 形式を使用することができますが、派生イメージには `raw` 形式を使用することができません。これは、`raw` 形式は `backing_file` オプションに対応していないためです。下記の例では、派生イメージに `qcow2` 形式を使用します。

たとえば `/images/sles_base.raw` が `raw` 形式のベースイメージであるとします。

```
> qemu-img info /images/sles_base.raw
image: /images/sles_base.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 2.4G
```

イメージの予約サイズは 4 GB で実サイズは 2.4 GB 、そして形式が `raw` であることがわかります。あとは `/images/sles_base.raw` ファイルに対する派生イメージを作成します:

```
> qemu-img create -f qcow2 /images/sles_derived.qcow2 \
-o backing_file=/images/sles_base.raw
Formatting '/images/sles_derived.qcow2', fmt=qcow2 size=4294967296 \
backing_file='/images/sles_base.raw' encryption=off cluster_size=0
```

派生イメージの詳細を確認します:

```
> qemu-img info /images/sles_derived.qcow2
image: /images/sles_derived.qcow2
file format: qcow2
virtual size: 4.0G (4294967296 bytes)
disk size: 140K
cluster_size: 65536
backing file: /images/sles_base.raw \
(actual path: /images/sles_base.raw)
```

派生イメージの予約サイズはベースイメージと同じサイズ (4 GB) になっていますが、実サイズは 140 KB しかありません。これは、派生イメージにはベースイメージからの変更点のみを記録する仕組みであるためです。あとは派生イメージを利用して仮想マシンを起動し、必要に応じて登録などの処理を行い、最新の修正を適用してください。不要なパッケージを削除してもかまいませんし、必要な新しいパッケージをインストールしてもかまいません。あとは VM ゲスト をシャットダウンし、再度派生イメージの状態を確認してみます:

```
> qemu-img info /images/sles_derived.qcow2
image: /images/sles_derived.qcow2
file format: qcow2
virtual size: 4.0G (4294967296 bytes)
disk size: 1.1G
cluster_size: 65536
backing file: /images/sles_base.raw \
(actual path: /images/sles_base.raw)
```

上記の `disk size` の値にもあるとおり、この例では実サイズが 1.1 GB まで拡張しました。ここには、ベースイメージからのファイルシステム内の変更点が含まれています。

35.2.4.3 派生イメージの再ベース

派生イメージの修正 (修正の適用、必要なアプリケーションのインストール、各種の設定変更等々) を行うことで、必要な環境を構築することができます。ここからベースイメージをマージ (併合) することができるほか、新しいベースイメージを元に派生イメージを作り直すこともできます。

元のベースイメージ (例: `/images/sles_base.raw`) には、新規にインストールしたシステムが含まれています。このベースイメージを雛形として、最新のセキュリティ更新などをインストールして、新しいベースイメージを作成することができます。新しいベースイメージからも派生イメージを作成して、必要なアプリケーションやサービスなどを動作させることができます。なお、新しいベースイメージは元のベースイメージから独立させることができます。このような新しいベースイメージの作成を、再ベース と呼びます。

```
> qemu-img convert /images/sles_derived.qcow2 \
-O raw /images/sles_base2.raw
```

上記のコマンドを実行すると、`raw` 形式で新しいベースイメージ `/images/sles_base2.raw` を作成することができます。

```
> qemu-img info /images/sles_base2.raw
image: /images/sles11_base2.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 2.8G
```

新しいイメージは元のイメージに比べて 0.4 ギガバイトほど大きくなっています。このイメージファイルはベースイメージを使用していませんので、ここから新しい派生イメージを作成することができます。これにより、組織の要件にあわせ、より洗練された仮想ディスクイメージ階層を作成することで、時間と手間の両方を削減できるようになります。

35.2.4.4 VM ホストサーバ 側でのイメージのマウント

仮想ディスクイメージは、ホストシステム内でマウントすることもできます。なお、あらかじめ 第20章「libguestfs」を読むとともに、仮想マシンイメージにアクセスするための専用ツールを使用しておくことを強くお勧めします。ただし、どうしても手作業でこれを実施しなければならないような場合、本章をお読みのうえ作業を行ってください。

Linux システムでは、`raw` 形式のディスクイメージであれば、ループバックデバイスを利用して内部のパーティションをマウントすることができます。最初の例は複雑ではありますがわかりやすいもの、2 つ目の例は直感的なものになっています:

手順 35.1: パーティションオフセット値を指定したディスクイメージのマウント

1. まずはマウントしたいディスクイメージに対して loop デバイスを割り当てます。

```
> losetup /dev/loop0 /images/sles_base.raw
```

2. セクタサイズ の項目を探し、マウントしたいパーティションの セクタ番号 の開始位置を判断します。

```
> fdisk -lu /dev/loop0

ディスク /dev/loop0: 4294 MB, 4294967296 バイト
単位: セクタ (1 * 512 = 512 ❶バイト)
セクタサイズ (論理 / 物理): 512 バイト / 512 バイト
I/O サイズ (最小 / 推奨): 512 バイト / 512 バイト
ディスクラベルのタイプ: dos
ディスク識別子: 0x000ceca8

デバイス        起動    開始位置     終了位置      セクタ        Id  タイプ
/dev/loop0p1              63     1542239      771088+  82  Linux スワップ / Solaris
/dev/loop0p2   *     1542240❷     8385929     3421845   83  Linux
```

❶ ディスクのセクタサイズを表しています。

❷ パーティションのセクタ開始位置を表しています。

3. 開始位置を計算します:
   セクタサイズ * 開始位置_(セクタ) = 512 * 1542240 = 789626880

4. いったんループバックデバイスの割り当てを解除したあと、計算結果をもとに offset 値を指定して、特定のディレクトリにマウントします。

```
> losetup -d /dev/loop0
> mount -o loop,offset=789626880 \
/images/sles_base.raw /mnt/sles/
> ls -l /mnt/sles/
total 112
drwxr-xr-x   2 root root  4096 Nov 16 10:02 bin
drwxr-xr-x   3 root root  4096 Nov 16 10:27 boot
drwxr-xr-x   5 root root  4096 Nov 16 09:11 dev
[...]
drwxrwxrwt  14 root root  4096 Nov 24 09:50 tmp
drwxr-xr-x  12 root root  4096 Nov 16 09:16 usr
drwxr-xr-x  15 root root  4096 Nov 16 09:22 var
```

5. あとはマウントしたパーティションに対して必要な作業を実施し、完了したらマウントを解除します。

```
> cp /etc/X11/xorg.conf /mnt/sles/root/tmp
> ls -l /mnt/sles/root/tmp
> umount /mnt/sles/
```



警告: 使用中のイメージに対して書き込みを行ってはならない件について

動作中の仮想マシンが使用するパーティションを **読み書き可能な** モードでマウントしてはなりません。読み書き可能な状態でマウントしてしまうと、パーティションを破壊する結果になるほか、VM ゲスト 全体を破壊する結果になってしまいます。

36 qemu-system-ARCH を利用した仮想マシンの実行

改訂履歴

2026-08-20

現在の QEMU 書式に合わせて、ネットワークの設定例にて `-netdev` の行内に netdev のバックエンドオプション (例: `ifname` , `script` , `downscript` , `restrict` , `net` , `host` , `dhcpstart` , `hostname` , `tftp` , `bootfile` , `hostfwd`) を追加。廃止予定となった `vlan` サブオプションを削除し、同様に廃止予定となった `macaddr` を `mac` サブオプションに修正。

2024-06-27

仮想ディスクイメージの準備 (詳しくは 35.2項「`qemu-img` を利用したディスクイメージの管理」をお読みください) を行ったあとは、対応する仮想マシンの起動を行うことができます。35.1項「`qemu-system-ARCH` を利用した基本的なインストール」では VM ゲスト のインストールから実行までの簡単なコマンドを紹介してきましたが、本章では `qemu-system-ARCH` のより詳しい説明のほか、特定の作業を行うための解決方法についても説明しています。`qemu-system-ARCH` に用意されている全てのオプションについて説明を読みたい場合は、マニュアルページ (`man 1 qemu`) をご覧ください。

36.1 基本的な `qemu-system-ARCH` の実行方法

`qemu-system-ARCH` コマンドは下記の書式で実行します:

```
qemu-system-ARCH オプション❶ -drive file=ディスクイメージ❷
```

❶ `qemu-system-ARCH` には多数のオプションが存在しています。これらのうちのほとんどは擬似ハードウエアを定義するためのパラメータで、残りはエミュレータの一般的な動作に影響があるものです。何もオプションを指定しない場合は、いずれも既定値が適用されますが、少なくともディスクイメージについては指定する必要があります。

❷ ディスクイメージには、仮想化を行うゲストシステムのディスクのイメージが含まれているものを指定します。`qemu-system-ARCH` では多数のイメージ形式に対応していますので、詳しくは `qemu-img --help` と入力して実行してください。

重要: AArch64 アーキテクチャについて

KVM サポートは 64 ビット Arm® アーキテクチャ (AArch64) 向けにのみ提供されています。AArch64 アーキテクチャで QEMU を実行する場合、それぞれ下記を行う必要があります:

- `-machine virt-バージョン番号` オプションを指定して、QEMU Arm® 仮想マシンの種類を選択する必要があります。
- `-bios` オプションを指定して、ファームウエアイメージファイルを選択する必要があります。
  ファームウエアイメージファイルの指定に関しては、`-drive` オプションを利用する方法もあります。たとえば下記のようになります:

  ```
  -drive file=/usr/share/edk2/aarch64/QEMU_EFI-pflash.raw,if=pflash,format=raw
  -drive file=/var/lib/libvirt/qemu/nvram/opensuse_VARS.fd,if=pflash,format=raw
  ```

- `-cpu host` オプションを指定して、VM ホストサーバ の CPU を指定する必要があります (既定値: `cortex-15`) 。
- ホストと同一のバージョンの Generic Interrupt Controller (GIC) を使用したい場合は、`-machine gic-version=host` を指定する必要があります (既定値: `2`) 。
- グラフィックモードが必要な場合は、`virtio-gpu-pci` のグラフィックデバイスの種類を指定する必要があります。

たとえば下記のようになります:

```
> sudo qemu-system-aarch64 [...] \
 -bios /usr/share/qemu/qemu-uefi-aarch64.bin \
 -cpu host \
 -device virtio-gpu-pci \
 -machine virt,accel=kvm,gic-version=host
```

36.2 一般的な `qemu-system-ARCH` のオプション

本章では `qemu-system-ARCH` の一般的なオプションのほか、仮想マシンのプロセッサやメモリ、モデルや時刻処理方式など、基本的な仮想ハードウエアに関連するオプションを説明しています。

`-name ゲストの名前`

動作させるゲストシステムの名前を指定します。名前はウインドウのタイトル部分に表示されるほか、VNC サーバでも使用されます。

`-boot オプション`

起動時にどの順番でドライブを読み込むのかを指定します。ドライブはそれぞれ文字で表され、`a` と `b` がそれぞれフロッピィディスクドライブの 1 台目と 2 台目、`c` が 1 台目のハードディスク、`d` が 1 台目の CD-ROM ドライブ、そして `n` から `p` までがイーサネットアダプタかのネットワーク起動を表します。

たとえば `qemu-system-ARCH [...] -boot order=ndc` のように指定すると、最初にネットワークからの起動を試し、次に 1 台目のCD-ROM ドライブから、最後に 1 台目のハードディスクからの起動を試すことになります。

`-pidfile ファイル名`

QEMU のプロセス識別番号 (PID) のファイルへの保存先を指定します。これは QEMU をスクリプト内で動作させる場合に有用です。

`-nodefaults`

既定では、QEMU はコマンドラインから何も指定を行わない場合、基本的な仮想デバイスを作成します。このオプションを指定すると、その機能が無効化され、グラフィックカードやネットワークカード、パラレルポートやシリアルポートなど、それぞれのデバイスを個別に手作業で指定しなければならなくなります。また、この場合 QEMU モニタも既定では接続されなくなります。

`-daemonize`

QEMU を起動したあと、プロセスを「デーモン化」します。デーモン化を行うことで、仮想マシンの準備が整った段階で標準入出力から切り離されるようになります。



注記: SeaBIOS BIOS 実装について

SeaBIOS は既定で使用される BIOS です。USB デバイスのほか、任意のドライブ (CD-ROM ドライブ, フロッピィディスクドライブ, ハードディスクドライブ) から起動を行うことができます。また、USB マウスやキーボードにも対応し、複数の VGA カードを接続することもできます。SeaBIOS に関する詳細は、SeaBIOS の Web サイト (https://www.seabios.org/SeaBIOS) ↗ (英語) をご覧ください。

36.2.1 基本的な仮想ハードウエア構成

36.2.1.1 マシンの種類

擬似するマシンの種類を指定することができます。対応するマシンの種類について、詳しくは `qemu-system-ARCH -M help` を実行して確認してください。



注記: ISA-PC について

isapc: ISA-only-PC の種類はサポート対象外となっております。

36.2.1.2 CPU モデル

プロセッサ (CPU) の型式を指定したい場合は、`qemu-system-ARCH -cpu 型式` のように指定して実行してください。指定可能な CPU の型式について、詳しくは `qemu-system-ARCH -cpu help` と入力して実行してください。

36.2.1.3 その他の基本オプション

下記に示すオプションは、コマンドラインから qemu を起動する場合によく使用されるオプションです。利用可能なオプションの一覧を表示したい場合は、qemu-doc のマニュアルページをお読みください。

`-m メガバイト`

VM ゲストに割り当てる RAM のサイズを、メガバイト単位で指定します。

`-balloon virtio`

VM ゲストに割り当てる RAM のサイズを動的に変更することのできる、準仮想化デバイスを指定します。動的な割り当ての上限は `-m` で指定します。

`-smp CPU_数`

VM ゲストに提供する擬似 CPU 数を指定します。QEMU では PC プラットフォームの場合、最大で 255 個 (KVM アクセラレーションを使用した場合 64 個) までの CPU を指定することができます。このオプションでは、ソケット数 やソケットあたりの コア数 、コアあたりの スレッド数 など、CPU 関連のパラメータを指定することもできます。

下記は `qemu-system-ARCH` のコマンドライン例です:

```
> sudo qemu-system-x86_64 \
```

```
-name "SLES 15.7" \
-M pc-i440fx-2.7 -m 512 \
-machine accel=kvm -cpu kvm64 -smp 2 \
-drive format=raw,file=/images/sles.raw
```



図 36.1: VM ゲスト として SLES を動作させた場合の QEMU のウインドウ

`-no-acpi`

ACPI サポートを無効化します。

`-S`

QEMU を CPU を停止させた状態で開始します。CPU を起動するには、QEMU モニタ内で `c` を押します。詳しくは 第37章「QEMU モニタを利用した仮想マシンの管理」をお読みください。

36.2.2 仮想デバイスの設定保存と読み込み

`-readconfig 設定ファイル`

VM ゲスト を起動する際、コマンドラインでデバイスの設定オプションを指定する代わりに、設定ファイル内にそれらを保存して使用することができます。設定ファイルの作成は `-writeconfig` で書き込むことができるほか、手作業で作成することもできます。

`-writeconfig 設定ファイル`

現在の仮想マシンのデバイス設定を、テキストファイルに保存します。保存したテキストファイルは、後から `-readconfig` オプションで指定することができます。

```
> sudo qemu-system-x86_64 -name "SLES 15.7" \
 -machine accel=kvm -M pc-i440fx-2.7 -m 512 -cpu kvm64 \
 -smp 2 /images/sles.raw -writeconfig /images/sles.cfg
(exited)
> cat /images/sles.cfg
# qemu config file

[drive]
  index = "0"
  media = "disk"
  file = "/images/sles_base.raw"
```

このような仕組みにより、仮想マシンの設定をより効率的かつ整理して管理することができるようになります。

36.2.3 ゲスト側のリアルタイムクロック

`-rtc オプション`

VM ゲスト 内での RTC (リアルタイムクロック) の処理方法を指定します。既定ではゲスト側のクロックはホストシステム側から派生させて使用することになります。そのため、ホスト側のシステムクロックは、正確な外部クロック (例: NTP サービス) と同期させておくことをお勧めします。VM ゲスト と VM ホストサーバ のクロックを分離したい場合は、既定値である `clock=host` の代わりに `clock=vm` を指定してください。
なお、VM ゲスト 側の起動時点の時刻を指定することもできます。時刻の指定は `base` オプションで行ってください:

```
> sudo qemu-system-x86_64 [...] -rtc clock=vm,base=2010-12-03T01:02:00
```

上記のようなタイムスタンプではなく、`utc` や `localtime` を指定することもできます。前者の場合、VM ゲスト を現在の UTC 時刻 (Coordinated Universal Time (協定世界時; https://ja.wikipedia.org/wiki/UTC ↗)) に、後者の場合は現在のローカル時刻にそれぞれ設定されます。

36.3 QEMU 内でのデバイスの使用

QEMU の仮想マシンは VM ゲストを動作させるのに必要な全てのデバイスを擬似します。QEMU では、ネットワークカードやブロックデバイス (ハードディスク、リムーバブルドライブなど) 、USB デバイスやキャラクタデバイス (シリアルポートやパラレルポート) 、マルチメディアデバイス (グラフィックカード、サウンドカード) などに対応しています。本章では、対応するさまざまな種類のデバイスについて、それらを設定するオプションを紹介しています。



ヒント

なお、デバイス側 (例: `-drive`) で特殊なドライバやドライバ設定を指定する必要がある場合、ドライバは `-device` オプションで設定し、ドライバを `drive=` で指定してください。具体的には下記のようになります:

```
> sudo qemu-system-x86_64 [...] -drive if=none,id=drive0,format=raw \
-device virtio-blk-pci,drive=drive0,scsi=off ...
```

利用可能なドライバや設定に関する情報を得たい場合は、`-device ?` オプション、もしくは `-device ドライバ名,?` オプションを指定して実行してください。

36.3.1 ブロックデバイス

ブロックデバイスは仮想マシンを動作させるのに必須のデバイスです。一般的には固定メディアと取り外し可能メディアの 2 種類が存在し、これらを総称して ドライブ と呼んでいます。通常は、接続されているうちの 1 台にゲスト側のオペレーティングシステムがインストールされます。

仮想マシン のドライブは `-drive` で定義します。このオプションには多数のサブオプションが存在し、それらのうちのいくつかを本章で紹介しています。全ての一覧を読みたい場合は、マニュアルページ (`man 1 qemu`) をご覧ください。

`-drive` オプションに対するサブオプション

`file=ファイルパス`

: このドライブで使用するディスクイメージのパスを指定します。何も指定しない場合、何も挿入されていない (リムーバブル) ドライブであるものとされます。

`if=インターフェイス`

: ドライブが接続されているインターフェイスの種類を指定します。現時点では、SUSE では `floppy` (フロッピィディスク) , `scsi` (SCSI) , `ide` (IDE) , `virtio` (virtio) のみをサポート対象としています。`virtio` を指定すると、準仮想化型ディスクドライバを使用します。既定値は `ide` です。

`index=コネクタ番号`

ディスクインターフェイス (詳しくは `if` オプションをご覧ください) 側での、ディスクが接続されているコネクタの番号を指定します。何も指定しない場合、コネクタ番号は自動で付与されます。

`media=種類`

メディアの種類を指定します。ハードディスクの場合は `disk` を、リムーバブル CD-ROM ドライブの場合は `cdrom` を指定します。

`format=形式`

接続されているディスクイメージの形式を指定します。何も指定しない場合、形式を自動検出します。現時点では、SUSE では `raw` , `qcow2` をそれぞれサポートします。

`cache=キャッシュ方式`

ドライブに対するキャッシュ方法を指定します。指定可能な値は `unsafe` , `writethrough` , `writeback` , `directsync` , `none` のいずれかです。`qcow2` イメージ形式で性能を改善したい場合は `writeback` を指定してください。なお `none` を指定するとホスト側でのキャッシュが無効化されますので、最も安全な選択となります。イメージファイルの場合、既定値は `writeback` となります。詳しくは 第18章「ディスクのキャッシュモード」をお読みください。



ヒント

ブロックデバイスの設定を単純化することもできます。QEMU にはいくつかのショートカットが規定されていて、これらを使用することで `qemu-system-ARCH` のコマンドラインを短くすることができます。

たとえば下記のように指定することができます:

```
> sudo qemu-system-x86_64 -cdrom /images/cdrom.iso
```

上記は、下記のような意味になります:

```
> sudo qemu-system-x86_64 -drive format=raw,file=/images/
cdrom.iso,index=2,media=cdrom
```

また、下記のように指定することもできます:

```
> sudo qemu-system-x86_64 -hda /images/imagei1.raw -hdb /images/image2.raw -hdc \
/images/image3.raw -hdd /images/image4.raw
```

上記は、下記のような意味になります:

```
> sudo qemu-system-x86_64 -drive format=raw,file=/images/
image1.raw,index=0,media=disk \
```

```
-drive format=raw,file=/images/image2.raw,index=1,media=disk \
-drive format=raw,file=/images/image3.raw,index=2,media=disk \
-drive format=raw,file=/images/image4.raw,index=3,media=disk
```



ヒント: イメージではなくホストのドライブを使用する方法について

ディスクイメージを使用 (詳しくは 35.2項「`qemu-img` を利用したディスクイメージの管理」) する代わりに、VM ホストサーバ 内にある既存のディスク選択してドライブとして使用し、VM ゲスト からアクセスすることもできます。この場合は、ディスクイメージのファイルパスを指定する箇所で、ホスト側のデバイス名をそのまま記述してください。

たとえばホスト側の CD-ROM ドライブにアクセスするには、下記のように指定します:

```
> sudo qemu-system-x86_64 [...] -drive file=/dev/cdrom,media=cdrom
```

ホスト側のハードディスクにアクセスするには、下記のように指定します:

```
> sudo qemu-system-x86_64 [...] -drive file=/dev/hdb,media=disk
```

なお、VM ゲスト 側が使用しているホスト側のドライブは、VM ホストサーバ 側からアクセスしてはなりませんし、他の VM ゲスト からもアクセスしてはなりません。

36.3.1.1 未使用のゲスト側ディスク領域の解放

スパースイメージファイル はディスクイメージファイルの形式で、その中にデータが書き込まれていくことによって、そのディスクイメージに指定したサイズまで自動的に拡張が行われるものを指します。たとえばスパースディスクイメージ内に 1 GB のデータをコピーすると、そのイメージのサイズは 1 GB だけ拡張します。ですが、そのうちの 500 MB 分のデータを削除しても、既定ではイメージのサイズは自動的には縮小しません。

このような理由から、KVM のコマンドラインには `discard=on` というオプションが用意されるようになっています。これはハイパーバイザに対して、スパースイメージ内からデータが削除された場合、そのうちの「不要な」箇所を自動的に解放する設定です。なお、このオプションはドライブインターフェイスに `if=scsi` を指定した場合にのみ設定することができます:

```
> sudo qemu-system-x86_64 [...] -drive format=イメージ形式,file=ファイルイメージのパス,if=scsi,discard=on
```

重要: サポート状態について

`if=scsi` はサポート対象外となっております。このインターフェイスは virtio-scsi ではなく、lsi SCSI アダプタ に結びつけられているためです。

36.3.1.2 IOThreads

IOThreads は virtio デバイス向けのイベントループスレッドで、スケーラビリティを改善するために I/O 要求を専門で処理するスレッドです。特に SMP 型の VM ホストサーバ 内で多数のディスクデバイスの接続された SMP 型の VM ゲスト を動作させているような場合に有用です。I/O 処理に QEMU のメインイベントループを使用する代わりに、IOThreads では複数の CPU に I/O 処理を分散させ、適切に設定されていれば遅延を改善することができるようになっています。

IOThreads は IOThread オブジェクトを定義することで有効化することができます。あとは virtio デバイスを使用することで、I/O イベントループ向けのオブジェクトを使用するようになります。多くの virtio デバイスで 1 つの IOThread オブジェクトを共用することもできますし、virtio デバイスと IOThread オブジェクトを 1:1 で作成して割り当てることもできます。下記の例では、ID `iothread0` で IOThread を 1 つだけ作成し、それを 2 つの virtio-blk デバイスから使用する場合の例を示しています。

```
> sudo qemu-system-x86_64 [...] -object iothread,id=iothread0\
-drive if=none,id=drive0,cache=none,aio=native,\
format=raw,file=filename -device virtio-blk-pci,drive=drive0,scsi=off,\
iothread=iothread0 -drive if=none,id=drive1,cache=none,aio=native,\
format=raw,file=filename -device virtio-blk-pci,drive=drive1,scsi=off,\
iothread=iothread0 [...]
```

下記の qemu コマンドラインでは、virtio デバイスと IOThread を 1:1 で割り当てる設定を示しています:

```
> sudo qemu-system-x86_64 [...] -object iothread,id=iothread0\
-object iothread,id=iothread1 -drive if=none,id=drive0,cache=none,aio=native,\
format=raw,file=filename -device virtio-blk-pci,drive=drive0,scsi=off,\
iothread=iothread0 -drive if=none,id=drive1,cache=none,aio=native,\
format=raw,file=filename -device virtio-blk-pci,drive=drive1,scsi=off,\
    iothread=iothread1 [...]
```

36.3.1.3 virtio-blk 向けの bio ベースの I/O パス

I/O に負荷が集中する環境の性能を改善するため、カーネルバージョン 3.7 では、virtio-blk インターフェイスに対して新しい I/O パスが提供されるようになりました。この bio ベースのブロックデバイスドライバは I/O スケジューラを迂回して動作するため、ゲスト側の I/O パスを短くすることができますので、遅延を短くすることができるようになっています。これは特に、SSD ディスクなどの高速なストレージデバイスで有用です。

このドライバは既定では無効化されています。使用するには、下記を実施します:

1. ゲスト側のカーネルコマンドラインに対して、`virtio_blk.use_bio=1` を追加します。これは [YaST] › [システム] › [ブートローダ] から行うことができます。
   それ以外にも、`/etc/default/grub` ファイルを編集して、`GRUB_CMDLINE_LINUX_DEFAULT=` 以下に上記の内容をスペース区切りで追加してもかまいません。追加した後は、`grub2-mkconfig >/boot/grub2/grub.cfg` を実行すると、grub2 の起動メニューに反映させることができます。
2. あとはゲスト側のカーネルコマンドラインを反映させるため、再起動を行います。



ヒント: 遅いデバイスでの bio ベースドライバについて

bio ベースの virtio-blk ドライバは、ハードディスクドライブのような遅いデバイスの場合、あまり役には立ちません。これは、bio によるパスの短縮よりもスケジューリングの利点のほうが大きいためです。そのため、遅いデバイスでは bio ベースのドライバを使用してはなりません。

36.3.1.4 iSCSI リソースへの直接アクセス

QEMU は現在 `libiscsi` との統合が行われています。これにより、QEMU から iSCSI デバイスに直接アクセスすることができるようになっているほか、iSCSI デバイスを仮想マシンのブロックデバイスとして使用することもできるようになっています。この機能を使用する場合もホスト側の iSCSI イニシエータの設定は必要なく、libvirt iSCSI ターゲットベースのストレージプールで必要となるだけです。その代わり、QEMU は iSCSI のターゲット LUN に対して、ユーザスペースライブラリの libiscsi を利用して、直接アクセスすることになります。iSCSI ベースのディスクデバイスは、libvirt の XML 設定でも指定することができます。



注記: RAW イメージ形式

この機能は、iSCSI プロトコルにさまざまな技術的制限が存在することから、RAW イメージ形式を使用している場合にのみ使用することができます。

下記は iSCSI の接続に対する QEMU のコマンドラインインターフェイスを説明しています。



注記: virt-manager の制限事項について

libiscsi ベースのストレージのプロビジョニング機能は virt-manager インターフェイス側に公開されていませんが、ゲストの XML ファイルを直接編集することで設定を行うことができます。このような iSCSI ベースのストレージに対するアクセスは、コマンドラインから行います。

```
> sudo qemu-system-x86_64 -machine accel=kvm \
  -drive file=iscsi://192.168.100.1:3260/iqn.2016-08.com.example:314605ab-a88e-49af-
b4eb-664808a3443b/0,\
  format=raw,if=none,id=mydrive,cache=none \
  -device ide-hd,bus=ide.0,unit=0,drive=mydrive ...
```

下記は iSCSI プロトコルを使用したゲストドメインの XML ファイル例の抜粋です:

```
<devices>
...
  <disk type='network' device='disk'>
    <driver name='qemu' type='raw'/>
    <source protocol='iscsi' name='iqn.2013-07.com.example:iscsi-nopool/2'>
      <host name='example.com' port='3260'/>
    </source>
    <auth username='myuser'>
      <secret type='iscsi' usage='libvirtiscsi'/>
    </auth>
    <target dev='vda' bus='virtio'/>
  </disk>
</devices>
```

下記はホストベースの iSCSI イニシエータを使用する設定例ですが、上記との違いを比較してください:

```
<devices>
...
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='native'/>
    <source dev='/dev/disk/by-path/scsi-0:0:0:0'/>
    <target dev='hda' bus='ide'/>
    <address type='drive' controller='0' bus='0' target='0' unit='0'/>
  </disk>
  <controller type='ide' index='0'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
             function='0x1'/>
  </controller>
</devices>
```

36.3.1.5 QEMU での RADOS ブロックデバイスの使用

RADOS Block Devices (RBD) は Ceph クラスタ内にデータを保存する仕組みです。この仕組みではスナップショットやレプリケーション、データの一貫性維持などの機能が用意されています。お使いの KVM 管理下の VM ゲスト からは、他のブロックデバイスを使用する場合と同様の方法で使用することができます。

36.3.2 グラフィックデバイスとディスプレイのオプション

本章では擬似ビデオカードの種類選択のほか、VM ゲスト のグラフィカル出力へのアクセス方法に関わるオプションを説明しています。

36.3.2.1 ビデオカードの定義

QEMU では `-vga` オプションを使用して、VM ゲスト のグラフィカルな出力に使用するビデオカードの設定を行います。`-vga` オプションでは下記のような値を指定することができます:

`none`
: VM ゲスト のビデオカードを無効化します (ビデオカードの擬似を行いません) 。ただし、シリアルコンソール経由で動作中の VM ゲスト にアクセスすることはできます。

`std`
: 標準的な VESA 2.0 VBE ビデオカードを擬似します。VM ゲスト で高解像度のディスプレイを使用したい場合に選択してください。

qxl
: QXL は準仮想化グラフィックカードです。VGA との互換性もあります (つまり VESA 2.0 VBE に対応しています) 。`qxl` は、`spice` ビデオプロトコルを使用する場合、推奨されるグラフィックカードです。

virtio
: 準仮想化 VGA グラフィックカードです。

36.3.2.2 ディスプレイのオプション

下記のオプションは、VM ゲスト のグラフィカル出力の表示方法を制御するためのものです。

`-display gtk`

GTK ウインドウ内にビデオ出力を表示します。このインターフェイスには UI 要素が表示され、ここから動作中に VM の設定や制御を行うことができるようになります。

`-display sdl`

SDL を介してビデオ出力を行います。通常は個別のグラフィックウインドウで表示されます。詳しくは SDL のドキュメンテーションをお読みください。

`-spice オプション[,オプション[,...]]`

SPICE リモートデスクトッププロトコルを有効化します。

`-display vnc`

詳しくは 36.5項「VNC を利用した VM ゲスト の表示」をお読みください。

`-nographic`

QEMU のグラフィカル出力を無効化します。コンソールの内容は、擬似シリアルポートに転送されます。

`-nographic` を指定した仮想マシンを起動した場合、仮想コンソール内で `Ctrl` – `A` `H` を押すことで、さまざまなショートカットの一覧を表示することができます。これにはたとえば、コンソールと QEMU モニタの切り替えなどの機能があります。

```
> sudo qemu-system-x86_64 -hda /images/sles_base.raw -nographic

C-a h    print this help
C-a x    exit emulator
C-a s    save disk data back to file (if -snapshot)
C-a t    toggle console timestamps
C-a b    send break (magic sysrq)
C-a c    switch between console and monitor
C-a C-a  sends C-a
(pressed C-a c)

QEMU 2.3.1 monitor - type 'help' for more information
(qemu)
```

`-no-frame`

QEMU のウインドウに対する装飾を無効化します。専用のデスクトップ作業領域を用意するような場合に便利です。

`-full-screen`

QEMU のグラフィカル出力をフルスクリーンで表示します。

`-no-quit`

QEMU のウインドウ内にある閉じるボタンを無効化し、強制的に閉じられてしまうことを防ぎます。

`-alt-grab` , `-ctrl-grab`

既定では、QEMU のウインドウ内でマウスのボタンを押すと、 `Ctrl` – `Alt` を押すまではマウスカーソルが外に出られなくなります。このキーの組み合わせを変更し、 `Ctrl` – `Alt` – `Shift` (`-alt-grab`) もしくは右 `Ctrl` キー (`-ctrl-grab`) に変更することができます。

36.3.3 USB デバイス

KVM 内の VM ゲスト で利用可能な USB デバイスを作成する方法には 2 つのものがあります。1 つは VM ゲスト 内に新しい擬似 USB デバイスを作成する方法、もう 1 つは既存の USB デバイスを VM ゲスト に割り当てる方法です。QEMU 内で USB デバイスを使用するには、まず `-usb` オプションで汎用 USB ドライバを有効化する必要があります。あとは `-usbdevice` オプションで個別のデバイスを指定していきます。

36.3.3.1 VM ゲスト 内での USB デバイスの擬似

SUSE では現在、下記の種類の USB デバイスをサポートしています: `disk` , `host` , `serial` , `braille` , `net` , `mouse` , `tablet`

`-usbdevice` オプションで指定することのできる USB デバイスの種類

`disk`

ファイルをベースにしたマスストレージデバイスを擬似します。ファイルの形式を自動検出させたくない場合は、 `format` オプションで形式を指定してください。

```
> sudo qemu-system-x86_64 [...] -usbdevice
       disk:format=raw:/virt/usb_disk.raw
```

`host`

ホスト側のデバイスをパススルーします (`バス.アドレス` で識別します)

`serial`

ホスト側のキャラクタデバイスに対するシリアルコンバータを定義します。

`braille`

BrlAPI を使用してブライユ点字ディスプレイを擬似し、ブライユ点字の出力を行います。

`net`

CDC イーサネットおよび RNDIS プロトコルに対応するネットワークアダプタを擬似します。

`mouse`

仮想 USB マウスを擬似します。このオプションは既定の PS/2 マウスの擬似を上書きする設定です。`qemu-system-ARCH [...] -usbdevice mouse` を指定して起動した VM ゲスト 内で下記のように実行すると、ハードウエアの状態を表示することができます:

```
> sudo hwinfo --mouse
20: USB 00.0: 10503 USB Mouse
[Created at usb.122]
UDI: /org/freedesktop/Hal/devices/usb_device_627_1_1_if0
[...]
Hardware Class: mouse
Model: "Adomax QEMU USB Mouse"
Hotplug: USB
Vendor: usb 0x0627 "Adomax Technology Co., Ltd"
Device: usb 0x0001 "QEMU USB Mouse"
[...]
```

`tablet`

絶対座標系を使用するポインタデバイス (例: タッチスクリーン) を擬似します。このオプションは、既定の PS/2 マウスの擬似を上書きする設定となります。タブレットデバイスは、VM ゲストを VNC プロトコル経由で操作する場合に有用です。詳しくは 36.5項「VNC を利用した VM ゲスト の表示」をお読みください。

36.3.4 キャラクタデバイス

`-chardev` オプションを指定することで、新しいキャラクタデバイスを作成することができます。一般的には下記のような書式を使用します:

```
qemu-system-x86_64 [...] -chardev バックエンドの種類,id=ID_文字列
```

ここで、*バックエンドの種類* には、`null`, `socket`, `udp`, `msmouse`, `vc`, `file`, `pipe`, `console`, `serial`, `pty`, `stdio`, `braille`, `tty`, `parport` のいずれかを指定します。また、それぞれのキャラクタデバイスに対しては、最大で 127 文字までの識別 (ID) 文字列を指定しなければなりません。この識別文字列は、関連する他のディレクティブを指定する際に使用します。それぞれのバックエンドに対する全てのオプションについて、詳しくはマニュアルページ (`man 1 qemu`) をお読みください。下記にそれぞれの **バックエンド** に対する概要説明を記述します:

`null`

何もデータを出力せず、入力されたデータをそのまま廃棄するだけのデバイスを作成します。

`stdio`

QEMU プロセスの標準入出力に結びつけるデバイスを作成します。

`socket`

双方向のストリームソケットを作成します。`path` サブオプションを指定すると、Unix ソケットを作成します:

```
> sudo qemu-system-x86_64 [...] -chardev \
socket,id=unix_socket1,path=/tmp/unix_socket1,server
```

`server` サブオプションを指定すると、作成するソケットを待ち受け状態にします。`port` を指定すると、TCP ソケットを作成します:

```
> sudo qemu-system-x86_64 [...] -chardev \
socket,id=tcp_socket1,host=localhost,port=7777,server,nowait
```

このコマンドは、TCP ポートの 7777 番を利用して、待ち受け側の（ `server` ）ソケットを作成します。QEMU では待ち受けるポートに対して、クライアント側からの接続を待機しないようにしています（ `nowait` ）。

`udp`

VM ゲスト からの全てのネットワークトラフィックを、UDP プロトコルを介して指定したリモートホストに転送します。

```
> sudo qemu-system-x86_64 [...] \
-chardev udp,id=udp_fwd,host=mercury.example.com,port=7777
```

このコマンドは、VM ゲスト から受信したデータを、ポート 7777 から mercury.example.com 宛に送信します。

`vc`

新しい QEMU のテキストコンソールを作成します。下記のようにすることで、仮想コンソールの幅と高さを指定することができます:

```
> sudo qemu-system-x86_64 [...] -chardev vc,id=vc1,width=640,height=480 \
-mon chardev=vc1
```

このコマンドは `vc1` という名前の仮想コンソールを指定したサイズで作成し、QEMU モニタに接続します。

`file`

VM ゲスト からの全てのデータを VM ホストサーバ 内のファイルに書き込みます。`path` サブオプションを指定する必要があります。指定したファイルが存在しない場合は、作成されます。

```
> sudo qemu-system-x86_64 [...] \
-chardev file,id=qemu_log1,path=/var/log/qemu/guest1.log
```

既定では、QEMU はシリアルポートとパラレルポートに対応するキャラクタデバイスを作成するほか、QEMU モニタ向けに特殊なコンソールを作成します。必要であれば、独自のキャラクタデバイスを作成して上述の用途で使用することができます。この場合、下記のオプションを指定しておくと良いでしょう:

`-serial` キャラクタデバイス

上記のように指定することで、VM ゲスト の仮想シリアルポートが VM ホストサーバ 内の キャラクタデバイス に転送されるようになります。グラフィカルモードの場合、既定では仮想コンソール (`vc`) になりますし、グラフィカルモードでない場合は `stdio` (標準入出力) になります。このほか、`-serial` オプションには多数のサブオプションが用意されています。詳しくは `man 1 qemu` で表示することのできるマニュアルページをお読みください。

なお、最大で 4 個までのシリアルポートを擬似することができます。シリアルポートを全て無効化したい場合は、`-serial none` を指定してください。

`-parallel` デバイス

上記のように指定することで、VM ゲスト の仮想パラレルポートが VM ホストサーバ 内の デバイス に転送されるようになります。指定することのできるデバイスは `-serial` と同じです。



ヒント

VM ホストサーバ に openSUSE Leap を使用している場合、ハードウエア側のパラレルポートに直接接続することもできます。この場合は `/dev/parportN` (`N` はポート番号) のように指定してください。

なお、最大で 3 個までのパラレルポートを擬似することができます。パラレルポートを全て無効化したい場合は、`-parallel none` を指定してください。

`-monitor` キャラクタデバイス

QEMU のモニタを VM ホストサーバ 内の キャラクタデバイス に転送するよう設定します。指定することのできるデバイスは `-serial` と同じです。グラフィカルモードの場合、既定では仮想コンソール (`vc`) になりますし、グラフィカルモードでない場合は `stdio` (標準入出力) になります。

利用可能な全てのデバイスバックエンドについて、詳しくは (`man 1 qemu`) で表示することのできるマニュアルページをお読みください。

36.4 QEMU でのネットワーク

`-device` とともに `-netdev` オプションを指定することで、ネットワークの種類を指定して VM ゲストのネットワークインターフェイスを追加することができるようになります。`-netdev` オプションの書式は下記のとおりです:

```
-netdev 種類[,キー[=値][,...]]
```

現時点では、SUSE は `user`、`bridge`、`tap` の種類のみをサポート対象としています。`-netdev` で指定可能な全サブオプションの一覧について、詳しくはマニュアルページ（`man 1 qemu`）をお読みください。

対応する -netdev サブオプション

`bridge`
: 指定したネットワークヘルパーを使用して TAP インターフェイスの設定を行い、それを指定したブリッジに接続します。詳しくは 36.4.3項「ブリッジ型ネットワーク」をお読みください。

`user`
: ユーザモードネットワーキングを指定します。詳しくは 36.4.2項「ユーザモードネットワーク」をお読みください。

`tap`
: ブリッジもしくはルーティング型のネットワークを指定します。詳しくは 36.4.3項「ブリッジ型ネットワーク」をお読みください。

36.4.1 ネットワークインターフェイスカードの定義

`-device` オプションとともに `-netdev` オプションを指定することで、新しい擬似ネットワークカードを追加することができます:

```
> sudo qemu-system-x86_64 [...] \
-netdev tap❶,id=hostnet0 \
-device virtio-net-pci❷,netdev=hostnet0,\
mac=00:16:35:AF:94:4B❸
```

❶ ネットワークデバイスの種類を指定しています。

❷ ネットワークカードの型式を指定しています。お使いのプラットフォームの QEMU で指定可能な全てのネットワークカードの一覧を取得するには、`qemu-system-ARCH -device help` を実行して `Network devices:` セクション以下をご覧ください。

現時点では、SUSE は `rtl8139`，`e1000` およびその派生系である `e1000-82540em`，`e1000-82544gc`，`e1000-82545em`，`virtio-net-pci` にそれぞれ対応しています。そのドライバで指定可能なオプションの一覧を取得したい場合は、ドライバのオプションの箇所に `help` を追加してください:

```
> sudo qemu-system-x86_64 -device e1000,help
e1000 options:
  acpi-index=<uint32>    -  (default: 0)
  addr=<str>             - Slot and optional function number, example: 06.0 or 06
 (default: -1)
  bootindex=<int32>
  busnr=<busnr>
  extra_mac_registers=<bool> - on/off (default: on)
  failover_pair_id=<str>
  init-vet=<bool>        - on/off (default: on)
  mac=<str>              - Ethernet 6-byte MAC Address, example: 52:54:00:12:34:56
  migrate_tso_props=<bool> - on/off (default: on)
  multifunction=<bool>   - on/off (default: off)
  netdev=<str>           - ID of a netdev to use as a backend
  rombar=<int32>         -  (default: -1)
  romfile=<str>
  romsize=<uint32>       -  (default: 4294967295)
  x-max-bounce-buffer-size=<size> - Maximum buffer size allocated for bounce buffers
 used for mapped access to indirect DMA memory (default: 4096)
  x-pcie-ari-nextfn-1=<bool> - on/off (default: off)
  x-pcie-err-unc-mask=<bool> - on/off (default: on)
  x-pcie-ext-tag=<bool>  - on/off (default: on)
  x-pcie-extcap-init=<bool> - on/off (default: on)
  x-pcie-lnksta-dllla=<bool> - on/off (default: on)
```

❸ ネットワークカードの Media Access Control (MAC) アドレスを指定しています。この値はネットワーク内で唯一のものでなければならないため、必ず指定しておくことをお勧めします。指定しない場合、QEMU は既定の MAC アドレスを使用してしまうため、同じ VLAN 内で MAC アドレスの重複を引き起こしてしまうことになります。

36.4.2 ユーザモードネットワーク

`-netdev user` オプションを指定することで、QEMU に対してユーザモードのネットワーキングを使用するように指定することができます。これはネットワーキングモードを指定しない場合の既定であるため、下記のコマンドはいずれも同じ意味になります:

```
> sudo qemu-system-x86_64 -hda /images/sles_base.raw
```

```
> sudo qemu-system-x86_64 -hda /images/sles_base.raw -netdev user,id=hostnet0
```

このモードは、VM ゲスト からインターネットなどの外部リソースにアクセスさせる必要があるような場合に有用です。既定では着信側のトラフィックは許可されない仕組みであるため、VM ゲスト はネットワーク内では他のマシンから見えない存在になります。また、このネットワークモードを使用する場合、管理者権限は不要となります。ユーザモードは、VM ホストサーバ 内のローカルディレクトリから VM ゲスト をネットワークブートするような場合にも有用です。

なお、このモードの場合、仮想的な DHCP サーバを利用して VM ゲスト に IP アドレスが割り当てられます。VM ホストサーバ (DHCP サーバ) は 10.0.2.2 、そして割り当てられる IP アドレスの範囲は 10.0.2.15 以降となります。`ssh` や `scp` などで 10.0.2.2 を指定することで、VM ホストサーバ に接続してファイルのやり取りを行ったりすることができます。

36.4.2.1 コマンドライン例

本章ではユーザモードのネットワーキングの設定例をいくつか示しています:

例 36.1: 制限付きユーザモードネットワーク

```
> sudo qemu-system-x86_64 [...] \
-netdev user❶,id=hostnet0❷,restrict=yes❸ \
-device virtio-net-pci,netdev=hostnet0
```

❶ ユーザモードネットワーキングの指定を行っています。

❷ ネットワークバックエンドに対する分かりやすい名前を指定しています。QEMU モニタなどで識別する際に有用です。

❸ VM ゲスト を孤立させる指定です。これにより VM ホストサーバ と通信ができなくなるほか、外部ネットワークにもパケットが送信されなくなります。

例 36.2: 独自の IP アドレス範囲を指定したユーザモードネットワーク

```
> sudo qemu-system-x86_64 [...] \
-netdev user,id=hostnet0,net=10.2.0.0/8❶,host=10.2.0.6❷,\
dhcpstart=10.2.0.20❸,hostname=tux_kvm_guest❹ \
-device virtio-net-pci,netdev=hostnet0
```

❶ VM ゲスト との間で使用されるネットワークの IP アドレスとネットマスク (オプション) を指定しています。既定値は 10.0.2.0/8 です。

❷ VM ゲスト から見た VM ホストサーバ のアドレスを指定しています。既定値は 10.0.2.2 です。

❸ 内蔵の DHCP サーバが割り当てる最初の IP アドレスを指定しています。このアドレスから最大で 16 個までを割り当てることができます。既定値は 10.0.2.15 です。

❹ 内蔵 DHCP サーバが VM ゲスト に対して送信する、VM ゲスト 側のホスト名を指定しています。

例 36.3: ネットワーク起動と TFTP を利用したユーザモードネットワーク

```
> sudo qemu-system-x86_64 [...] \
-netdev user,id=hostnet0,tftp=/images/tftp_dir❶,\
bootfile=/images/boot/pxelinux.0❷ \
-device virtio-net-pci,netdev=hostnet0
```

❶ 内蔵 TFTP (非常に基本的な機能に限定されたファイル転送プロトコル) サーバを有効化しています。VM ゲスト 側からは、指定したディレクトリがルートディレクトリとなります。

❷ 指定したファイルを BOOTP (IP アドレスと起動イメージの場所を提供するネットワークプロトコル、主にディスクレスマシンで使用します) ファイルとして提供する指定です。`tftp` オプションとともに使用した場合、VM ゲスト はホスト内のローカルディレクトリ内のファイルからネットワーク起動できるようになります。

例 36.4: ホスト側でのポート転送を指定したユーザモードネットワーク

```
> sudo qemu-system-x86_64 [...] \
-netdev user,id=hostnet0,hostfwd=tcp::2222-:22 \
-device virtio-net-pci,netdev=hostnet0
```

ホスト側で TCP ポート番号 2222 に届いたパケットを、VM ゲスト 側の TCP ポート番号 22 (`SSH`) に転送する指定です。VM ゲスト 側で `sshd` が動作していれば、

```
> ssh ホスト -p 2222
```

のように入力することで、VM ゲスト 側の `SSH` に接続することができるようになります。ここで `ホスト` には VM ホストサーバ のホスト名または IP アドレスを指定します。

36.4.3 ブリッジ型ネットワーク

`-netdev tap` オプションを指定することで、QEMU はネットワークブリッジを作成し、ホスト側の TAP ネットワークデバイスを VM ゲスト の指定 VLAN に接続することができます。これにより、ネットワークインターフェイスはネットワーク側からアクセスできるようになります。なお、この方式は既定では選択されないため、明示的に指定する必要があります。

まずはネットワークブリッジを作成し、VM ホストサーバ の物理ネットワークインターフェイス (通常は `eth0`) をブリッジに追加します:

1. [YaST コントロールセンター] を起動し、[システム] › [ネットワークの設定] を選択します。
2. [追加] を押したあと、[ハードウエアダイアログ] ウインドウ内で、[デバイスの種類] のドロップダウンボックスで [ブリッジ] を選択します。あとは [次へ] を押します。
3. IP アドレスを動的もしくは静的に設定したい場合はそれぞれの項目を設定します。また、必要であればその他の設定も行います。

4. ［ブリッジ接続デバイス］のタブを選択して、ブリッジに追加するイーサネットデバイスを選択します。
   ［次へ］を行います。既に設定されているデバイスを追加する旨のメッセージが表示された場合は、［続行］を押します。
5. 最後に［OK］を押して変更点を保存します。あとはブリッジが作成されていることを確認します:

```
> bridge link
2: eth0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 \
 state forwarding priority 32 cost 100
```

36.4.3.1 ブリッジへの手動接続

下記のようなサンプルスクリプトを使用することで、VM ゲスト を新しく作成したブリッジ `br0` に接続することができます。スクリプト内のコマンドはいずれも管理者権限が必要となることから、`sudo` の仕組みを利用して実行するよう作られています。



ヒント: 必要なソフトウエアについて

ネットワークブリッジを管理するには、`tunctl` パッケージをインストールしておく必要があります。

```
#!/bin/bash
bridge=br0❶
tap=$(sudo tunctl -u $(whoami) -b)❷
sudo ip link set $tap up❸
sleep 1s❹
sudo ip link add name $bridge type bridge
sudo ip link set $bridge up
sudo ip link set $tap master $bridge❺
qemu-system-x86_64 -machine accel=kvm -m 512 -hda /images/sles_base.raw \
 -netdev tap,id=hostnet0,ifname=$tap❻,script=no❼,downscript=no \
 -device virtio-net-pci,netdev=hostnet0,mac=00:16:35:AF:94:4B
sudo ip link set $tap nomaster❽
sudo ip link set $tap down❾
sudo tunctl -d $tap❿
```

❶ ブリッジデバイスの名前です。

❷ 新しい TAP デバイスを作成し、スクリプトを実行したユーザに権限を与えます。TAP デバイスは仮想的なネットワークデバイスで、しばしば仮想化やエミュレーションで用いられるものです。

❸ 新しく作成した TAP ネットワークインターフェイスを起動しています。

❹ TAP ネットワークデバイスの準備を待つため、1 秒間の一時停止を設定しています。

5. 新しい `TAP` デバイスをネットワークブリッジ `br0` に追加しています。
6. `ifname=` サブオプションでは、ブリッジに使用する TAP ネットワークインターフェイスの名前を指定しています。
7. `qemu-system-ARCH` がネットワークブリッジに接続する前に、それぞれ `script` と `downscript` の値が調べられます。VM ホストサーバ のファイルシステム内に該当するスクリプトが見つかった場合、`script` はネットワークブリッジへの接続前に、`downscript` はネットワーク環境の終了時にそれぞれ実行されます。また、既定では `/etc/qemu-ifup` および `/etc/qemu-ifdown` をそれぞれ実行します。`script=no` および `downscript=no` を指定すると、これらのスクリプトは無効化され、必要に応じて手作業で作業を行うことになります。
8. ネットワークブリッジ `br0` から TAP インターフェイスを削除しています。
9. TAP デバイスの状態を `down` (停止) に設定しています。
10. TAP デバイスを削除しています。

36.4.3.2 qemu-bridge-helper を利用したブリッジへの接続

VM ゲスト をネットワークブリッジ経由でネットワークに接続する場合、もう 1 つの方法として `qemu-bridge-helper` というヘルパープログラムを使用する方法が用意されています。これは TAP インターフェイスを作成して指定したブリッジに接続するまでの処理を行います。既定のヘルパープログラムは `/usr/lib/qemu-bridge-helper` にあります。なお、このヘルパープログラムは setuid root ビットが設定されていて、かつ仮想化グループ (`kvm`) のメンバーのみに実行権限が与えられていますので、`qemu-system-ARCH` コマンド自身を `root` 権限で動作させる必要はありません。

ヘルパーは下記のようにしてネットワークブリッジを指定した場合、自動的に呼び出されます:

```
qemu-system-x86_64 [...] \
 -netdev bridge,id=hostnet0,br=br0 \
 -device virtio-net-pci,netdev=hostnet0
```

TAP デバイスの設定／設定解除を行うプログラムを用意することで、独自のプログラムを実行することもできます。ヘルパープログラムを指定するには、`helper=ヘルパープログラムのパス` のように指定してください:

```
qemu-system-x86_64 [...] \
 -netdev bridge,id=hostnet0,br=br0,helper=/path/to/bridge-helper \
 -device virtio-net-pci,netdev=hostnet0
```



ヒント

`qemu-bridge-helper` に対してアクセス権限を設定するには、`/etc/qemu/bridge.conf` ファイルを作成してください。たとえば下記のようなディレクティブを設定したものとすると、

```
allow br0
```

`qemu-system-ARCH` コマンドが VM ゲストをネットワークブリッジに追加する際、`br0` を指定できるようになります。

36.5 VNC を利用した VM ゲストの表示

既定の QEMU では GTK (クロスプラットフォーム型のツールキットライブラリ) ウインドウを作成し、VM ゲストのグラフィカルな出力を表示します。`-vnc` オプションを指定すると、QEMU は VNC ディスプレイを作成して接続を待ち受けるようになり、その VNC ディスプレイ内にグラフィカルな出力を行うようになります。



ヒント

VNC セッションで QEMU の仮想マシンを操作する場合、`-usbdevice tablet` オプションを指定しておくとよいでしょう。

これに加えて、`en-us` 以外のキーボードレイアウトを使用したい場合は、`-k` オプションで指定してください。

`-vnc` の最初のサブオプションには、*ディスプレイ* 番号を指定しなければなりません。`-vnc` オプションでは、下記のような形式でディスプレイ番号を指定します:

ホスト:*ディスプレイ*

ディスプレイ のディスプレイへの接続を、*ホスト* で指定したホストからの接続のみに制限します。VNC セッションで使用する TCP ポート番号は通常、5900 + *ディスプレイ* になります。また、*ホスト* を指定しない場合、任意のマシンから接続できるようになります。

`unix`:*パス*

Unix ドメインソケットを利用して VNC サーバの待ち受けを行います。*パス* には Unix ソケットの場所を指定します。

`none`

VNC サーバの機能を準備しますが、サーバ自身の開始は行わないようにします。この場合、QEMU モニタから開始することができます。詳しくは 第37章「QEMU モニタを利用した仮想マシンの管理」をお読みください。

ディスプレイ値の後にカンマ区切りで下記のような追加オプションを指定することができます:

`reverse`
: 逆 接続と呼ばれる方式を利用して、VNC クライアントに接続を行います。

`websocket`
: VNC Websocket 接続専用に追加の TCP 待ち受けポートを用意します。VNC WebSocket では、5700 + ディスプレイ の TCP ポートを使用します。

`password`
: クライアントから接続があった場合、パスワードベースの認証を求めます。

`tls`
: VNC サーバと通信を行う際、クライアント側で TLS を利用するよう求めます。

`x509=証明書ディレクトリ`
: `tls` を指定した場合にのみ使用されます。TLS セッションのネゴシエーションに使用する x509 の視覚情報を求めるようにします。

`x509verify=証明書ディレクトリ`
: `tls` を指定した場合にのみ使用されます。TLS セッションのネゴシエーションに使用する x509 の視覚情報を求めるようにします。

`sasl`
: クライアントが VNC サーバに接続する際、SASL による認証を求めるようにします。

`acl`
: x509 クライアント証明書や SASL のパーティを確認する際、アクセス制御リストの機能を有効化します。

`lossy`
: 有損失圧縮 (gradient, JPEG など) を使用するようにします。

`non-adaptive`
: 適応符号化機能を無効化します。既定では適応符号化機能は有効化されています。

`share=[allow-exclusive|force-shared|ignore]`
: ディスプレイの共有ポリシーを設定します。



注記

ディスプレイオプションについてさらに詳しく知るには、qemu-doc のマニュアルページをお読みください。

VNC の使用例:

```
tux > sudo qemu-system-x86_64 [...] -vnc :5
# (クライアント側で:)
wilber > vncviewer venus:5 &
```

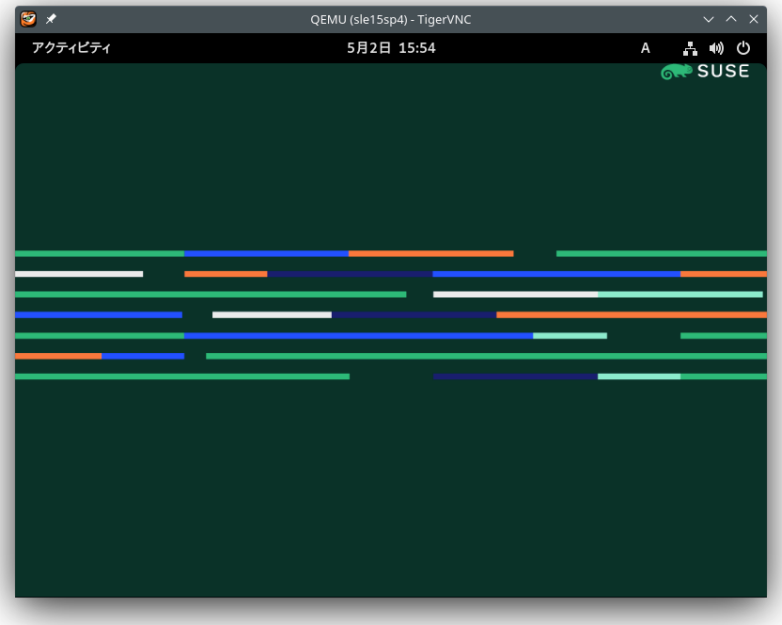


図 36.2: QEMU VNC セッション

36.5.1 VNC 接続の認証設定

既定の VNC サーバでは、何も認証を求めません。ここまでの例で設定を行うと、ネットワーク内の任意のマシンから QEMU の VNC セッションに接続し、画面を表示して操作ができるようになってしまいます。

VNC のクライアントとサーバの間で認証を必要とするよう設定したい場合、さまざまなレベルでそれを設定することができます。単純にパスワードのみで保護することができるほか、x509 証明書認証や SASL 認証を設定したり、これらを組み合わせて使用したりすることもできます。いずれも QEMU のコマンド 1 つで実現することができます。

VM ホストサーバ とクライアントの間での x509 認証の設定方法については、11.3.2項「x509 証明書を利用したリモート TLS/SSL 接続 (qemu+tls もしくは xen+tls)」と 11.3.2.3項「クライアントの設定とテスト」をそれぞれお読みください。

Remmina VNC ビューアでは高度な認証機構に対応しているため、下記の例に従って仮想マシンを起動し、Remmina で接続することで、VM ゲスト のグラフィカル出力を表示することができます。なお下記の例では、ホスト内の `/etc/pki/qemu` ディレクトリ内に、`ca-cert.pem` (証明機関の証明書), `server-cert.pem` (サーバの証明書), `server-key.pem` (サーバの機密鍵) の各ファイルが配置されているものとします。クライアント側では Remmina の起動時に設定を行うことができることから、任意のディレクトリに証明書を配置して使用することができます。

例 36.5: パスワード認証

```
qemu-system-x86_64 [...] -vnc :5,password -monitor stdio
```

VM ゲスト のグラフィカル出力を VNC ディスプレイ番号 5 (通常は TCP ポート 5905) に出力します。`password` サブオプションは、シンプルなパスワードベースの認証を有効化するための設定です。既定ではパスワードが何も設定されないため、起動後に QEMU モニタから `change vnc password` コマンドでパスワードを設定する必要があります:

```
QEMU 2.3.1 monitor - type 'help' for more information
(qemu) change vnc password
Password: ****
```

ここでは `-monitor stdio` を指定していますが、これは入出力を転送せずに QEMU モニタを使用することができないことによるものです。

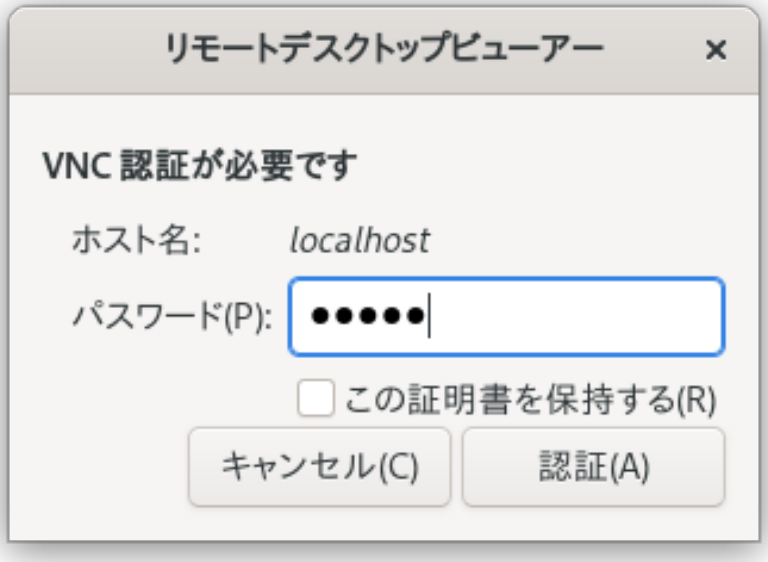


図 36.3: REMMINA での認証ダイアログ

例 36.6: X509 証明書認証

QEMU の VNC サーバが TLS 暗号化を行い、x509 の証明書を使用するように設定します。サーバはクライアントに対して証明書の提示を求め、証明機関の証明書に適合するかどうかを確認します。この認証方式を使用するには、内部用の証明機関を構築する必要があります。

```
qemu-system-x86_64 [...] -vnc :5,tls,x509verify=/etc/pki/qemu
```

例 36.7: X509 証明書とパスワードによる認証

TLS による暗号化と x509 証明書による認証に加えて、パスワード認証を追加で設定することもできます。これにより、クライアントに対して 2 階層の認証を設定することになります。なお、下記のコマンドを実行したあと、QEMU モニタでパスワードを忘れずに設定してください:

```
qemu-system-x86_64 [...] -vnc :5,password,tls,x509verify=/etc/pki/qemu \
-monitor stdio
```

例 36.8: SASL 認証

Simple Authentication and Security Layer (SASL) はインターネットプロトコル向けの認証およびデータセキュリティフレームワークです。PAM や Kerberos, LDAP など、さまざまな認証機構を組み合わせることができます。SASL では独自のユーザデータベースを使用する仕組みであるため、VM ホストサーバ 内にユーザアカウントを作成する必要はありません。

セキュリティ上の理由から、SASL 認証を使用する場合は、TLS 暗号化と x509 証明書認証を併用することをお勧めします:

```
qemu-system-x86_64 [...] -vnc :5,tls,x509,sasl -monitor stdio
```

37 QEMU モニタを利用した仮想マシンの管理

改訂履歴

2024-06-27

`qemu-system-アーキテクチャ` (たとえば `qemu-system-x86_64`) で仮想マシンを起動した場合、ユーザ側からの操作を受け付けるためのモニタコンソールが起動されます。モニタコンソール内ではコマンドを入力して実行することができますので、ここからリムーバブルメディアの変更やスクリーンショットの採取、オーディオのキャプチャなど、仮想マシンに対するさまざまな制御を行うことができます。



注記

下記の章では、便利な QEMU のモニタコマンドとその用途を説明しています。全てのコマンドを一覧で表示したい場合は、QEMU モニタのコマンドラインで `help` と入力して実行してください。

37.1 モニタコンソールへのアクセス



ヒント: `libvirt` 向けにはモニタコンソールが提供されない件について

モニタコンソールにアクセスするには、仮想マシンを `qemu-system-アーキテクチャ` コマンドで直接起動し、グラフィカルな出力を組み込みの QEMU ウインドウ内に表示させる必要があります。

仮想マシンを `libvirt` (たとえば `virt-manager`) で起動している場合で、VNC や Spice セッション経由で出力を表示させている場合は、モニタコンソールに直接アクセスすることはできません。ただし、`virsh` で下記のように入力して実行することで、モニタコマンドを送信することができます:

```
# virsh qemu-monitor-command コマンド
```

モニタコンソールへのアクセス方法は、仮想マシンの出力に使用しているディスプレイデバイスの種類によって異なります。ディスプレイに関する詳細は、36.3.2.2項「ディスプレイのオプション」をお読みください。たとえば `-display gtk` オプションを利用してモニタを表示させている場合、

Ctrl – Alt – 2 を押すことでモニタコンソールを表示させることができます。同様に -nographic オプションを利用している場合は、Ctrl – A C を押すことでモニタコンソールに切り替えることができます。

コンソールの使用方法についてヘルプを表示したい場合は、help もしくは ? と入力して実行します。特定のコマンドに対するヘルプを表示したい場合は、help コマンド のように入力して実行してください。

37.2 ゲストシステムに関する情報の取得

ゲストシステムに関する情報を取得したい場合は、info を使用します。オプションを何も指定しないで実行すると、指定可能なオプションの一覧が表示されます。オプションでは、表示したい情報を指定します:

`info version`
: QEMU のバージョンを表示します。

`info commands`
: 利用可能な QMP コマンドの一覧を表示します。

`info network`
: ネットワークの状態を表示します。

`info chardev`
: キャラクタデバイスを表示します。

`info block`
: ハードディスクやフロッピィディスク、CD-ROM ドライブなどのブロックデバイスに関する情報を表示します。

`info blockstats`
: ブロックデバイスに関する読み書きの統計情報を表示します。

`info registers`
: CPU レジスタを表示します。

`info cpus`
: 利用可能な CPU に関する情報を表示します。

`info history`
: コマンドラインの履歴を表示します。

`info irq`
: 輪の込みに関する統計情報を表示します。

`info pic`
: i8259 (PIC) の状態を表示します。

`info pci`
: PCI の情報を表示します。

`info tlb`
: 仮想メモリから物理メモリへのマッピング情報を表示します。

`info mem`
: 有効な仮想メモリマッピングを表示します。

`info jit`
: 動的なコンパイラの情報を表示します。

`info kvm`
: KVM の情報を表示します。

`info numa`
: NUMA の情報を表示します。

`info usb`
: ゲスト USB デバイスの情報を表示します。

`info usbhost`
: ホスト USB デバイスの情報を表示します。

`info profile`
: プロファイル情報を表示します。

`info capture`
: キャプチャ (オーディオ採取) 情報を表示します。

`info snapshots`
: 現時点で保存されている仮想マシンのスナップショットを表示します。

`info status`
: 現在の仮想マシンの状態に関する情報を表示します。

`info mice`
: どのマウスがイベントを受信しているのかを表示します。

`info vnc`
: VNC サーバの状態を表示します。

`info name`
: 現在の仮想マシンの名前を表示します。

`info uuid`
: 現在の仮想マシンの UUID を表示します。

`info usernet`
: ユーザネットワークスタックの接続情報を表示します。

`info migrate`
: 移行の状態を表示します。

`info balloon`
: バルーンデバイスの情報を表示します。

`info qtree`
: デバイスツリーを表示します。

`info qdm`
: qdev デバイスモデルリストを表示します。

`info roms`
: ROM を表示します。

`info migrate_cache_size`
: 現時点での移行 xbzrle (「Xor Based Zero Run Length Encoding」) のキャッシュサイズを表示します。

`info migrate_capabilities`
: xbzrle 圧縮などのさまざまな移行機能の状態を表示します。

`info mtree`
: VM ゲスト のメモリ階層構造を表示します。

`info trace-events`
: 利用可能なトレースイベントとその状態を表示します。

37.3 VNC パスワードの変更

VNC のパスワードを変更するには、`change vnc password` と入力して実行し、新しいパスワードを入力します:

```
(qemu) change vnc password
Password: ********
(qemu)
```

37.4 デバイスの管理

ゲストの動作中に新しいディスクを接続 (ホットプラグ) したい場合は、`drive_add` と `device_add` の各コマンドを使用します。まずは新しいドライブを定義し、それをバス (この例では 0) に接続する流れになります:

```
(qemu) drive_add 0 if=none,file=/tmp/test.img,format=raw,id=disk1
OK
```

ブロックサブシステムへの問い合わせを行うことで、新しいデバイスが追加されていることを確認することができます:

```
(qemu) info block
[...]
disk1: removable=1 locked=0 tray-open=0 file=/tmp/test.img ro=0 drv=raw \
encrypted=0 bps=0 bps_rd=0 bps_wr=0 iops=0 iops_rd=0 iops_wr=0
```

新しいドライブを定義したら、あとはゲスト側からアクセスすることができるようにするため、接続を行います。通常のデバイスであれば `virtio-blk-pci` もしくは `scsi-disk` のいずれかをドライバとして使用します。指定可能な値の一覧を表示するには、下記のように入力して実行します:

```
(qemu) device_add ?
name "VGA", bus PCI
name "usb-storage", bus usb-bus
[...]
name "virtio-blk-pci", bus virtio-bus
```

後は下記のようにしてデバイスを追加するだけです:

```
(qemu) device_add virtio-blk-pci,drive=disk1,id=myvirtio1
```

下記のように入力して実行すると、接続されたことを確認することができます:

```
(qemu) info pci
[...]
Bus  0, device   4, function 0:
```

```
    SCSI controller: PCI device 1af4:1001
      IRQ 0.
      BAR0: I/O at 0xffffffffffffffff [0x003e].
      BAR1: 32 bit memory at 0xffffffffffffffff [0x00000ffe].
      id "myvirtio1"
```



ヒント

`device_add` で追加したデバイスは、`device_del` で削除を行うことができます。詳しくは QEMU のモニタコマンドラインから `help device_del` と入力して実行することで表示される、ヘルプをお読みください。

リムーバブルデバイスのメディアを取り出すには、`eject` *デバイス名* コマンドを使用します。必要であれば `-f` オプションを追加して、強制的に取り出すこともできます。

リムーバブルメディア (たとえば CD-ROM) のメディアを交換したい場合は、`change` *デバイス名* コマンドを使用します。リムーバブルメディアの名前は、`info block` コマンドで確認することができます:

```
(qemu) info block
ide1-cd0: type=cdrom removable=1 locked=0 file=/dev/sr0 ro=1 drv=host_device
(qemu) change ide1-cd0 /path/to/image
```

37.5 キーボードとマウスの制御

モニタコンソールを使用することで、キーボードやマウスの入力を擬似することができます。たとえば、お使いのグラフィカルユーザインターフェイス側で認識されてしまうようなキー入力、たとえば X Window であれば `Ctrl` – `Alt` – `F1` を VM ゲストに送信したい場合、`sendkey` *キー入力* のように入力して実行することで、擬似的にキー入力を送信することができます:

```
sendkey ctrl-alt-f1
```

キー入力 の箇所で指定可能なキー名の一覧を表示するには、`sendkey` と入力して `<Tab>` を押します。

マウスを制御したい場合は、下記のようなコマンドを使用することができます:

`mouse_move` *DX dy* [*DZ*]

dx, dy (およびホイールスクロール dz) の分だけ、有効なマウスポインタを移動します。

`mouse_button` *値*

マウスボタンの押下状態を変更します (1=左, 2=中央, 4=右) 。

`mouse_set` インデックス

イベントを受信するマウスを選択します。デバイスのインデックス番号は、`info mice` コマンドで取得することができます。

37.6 利用可能なメモリの変更

仮想マシンを `-balloon virtio` オプション付きで起動している場合 (準仮想化デバイスが有効化されている場合) 、利用可能なメモリを動的に変更することができるようになります。バルーンデバイスの有効化に関する詳細については、35.1項「`qemu-system-ARCH` を利用した基本的なインストール」をお読みください。

モニタコンソール内でバルーンデバイスに関する情報を取得したり、バルーンデバイスが有効化されているかどうかを調べたりしたい場合は、`info balloon` コマンドを実行します:

```
(qemu) info balloon
```

バルーンデバイスが有効化されていれば、`balloon` メモリ量 (MB 単位) を入力して実行することで、メモリ量を変更することができます:

```
(qemu) balloon 400
```

37.7 仮想マシンのメモリダンプ

仮想マシンのメモリをディスクやコンソール出力に保存したい場合は、下記のコマンドをお使いください:

`memsave` アドレス サイズ ファイル名

アドレス で指定したアドレスを開始点として、サイズ で指定したサイズ分の仮想メモリダンプを ファイル名 のファイルに保存します。

`pmemsave` アドレス サイズ ファイル名

アドレス で指定したアドレスを開始点として、サイズ で指定したサイズ分の物理メモリダンプを ファイル名 のファイルに保存します。

x / 書式 アドレス

アドレス で指定したアドレスを開始点として、書式 文字列に従って仮想メモリダンプを出力します。このとき、書式 には カウント, 形式, サイズ をそれぞれ指定します:

カウント パラメータには表示すべき項目数を指定します。

形式 には x (16 進数), d (符号付き 10 進数), u (符号無し 10 進数), o (8 進数), c (char 型) or i (アセンブライン ストラクション) のいずれかを指定します。
サイズ パラメータには b (8 ビット), h (16 ビット), w (32 ビット) , g (64 ビット) のいずれかを指定します。x86 の場合、i で h や w を指定することで、16 ビットと 32 ビットのインストラクションサイズを選択することができます。

xp / 書式 アドレス
アドレス で指定したアドレスを開始点として、書式 文字列に従って物理メモリダンプを出力します。このとき、書式 には カウント , 形式 , サイズ をそれぞれ指定します:
カウント パラメータには表示すべき項目数を指定します。
形式 には x (16 進数), d (符号付き 10 進数), u (符号無し 10 進数), o (8 進数), c (char 型) or i (アセンブライン ストラクション) のいずれかを指定します。
サイズ パラメータには b (8 ビット), h (16 ビット), w (32 ビット) , g (64 ビット) のいずれかを指定します。x86 の場合、i で h や w を指定することで、16 ビットと 32 ビットのインストラクションサイズを選択することができます。

37.8 仮想コンソールのスナップショットの管理

QEMU モニタ内でのスナップショット管理機能は、SUSE ではサポートしていません。本章内での情報は、特定の用途で役に立つものです。

仮想マシン のスナップショット機能は、CPU やメモリ、書き込み可能な全てのディスクの内容を含む、仮想マシン内の全情報のスナップショットです。仮想マシンのスナップショット機能を使用するには、少なくとも 1 台以上のリムーバブルでない書き込み可能メディアが存在し、かつそれが qcow2 ディスク形式を使用していなければなりません。

スナップショット機能は、お使いの仮想マシンの状態を保存したい場合に便利な機能です。たとえば仮想サーバ内のネットワークサービスを設定していて、何か実験やテストなどを行ってサーバを不安定にしてしまうようなことを行いたい場合、あとから元の状態にすぐに安定状態に戻す用途で使用したりすることができます。また、仮想マシンの電源を落としてスナップショットを採取することで、バックアップとして使用することもできます。本章では前者について説明しています。後者については 35.2.3項「qemu-img を利用した仮想マシンのスナップショット管理」 で説明しています。

QEMU モニタ内でスナップショットを管理するコマンドとして、下記のようなものが用意されています:

savevm 名前
新しい仮想マシンのスナップショットを採取し、名前 で指定した名前で保存します。既に同名のスナップショットが存在している場合は、上書きされます。

`loadvm` 名前

名前 で指定した名前の仮想マシンスナップショットを読み込みます。

`delvm`

仮想マシンのスナップショットを削除します。

`info snapshots`

利用可能なスナップショットについて情報を表示します。

```
(qemu) info snapshots
Snapshot list:
ID❶       TAG❷                  VM SIZE❸    DATE❹           VM CLOCK❺
1         booting                 4.4M 2013-11-22 10:51:10   00:00:20.476
2         booted                  184M 2013-11-22 10:53:03   00:02:05.394
3         logged_in               273M 2013-11-22 11:00:25   00:04:34.843
4         ff_and_term_running     372M 2013-11-22 11:12:27   00:08:44.965
```

❶ スナップショットに対して自動的に割り当てられる識別番号です。通常は 1 ずつ加算されます。

❷ スナップショットの説明文字列です。ID を分かりやすく説明したものと言えます。

❸ スナップショットが占有しているディスク領域のサイズです。動作中のアプリケーションが多くのメモリを確保しているほど、スナップショットのサイズも大きくなります。

❹ スナップショットを採取した日時を表します。

❺ その時点での仮想マシンのクロック状態を表しています。

37.9 仮想マシンの一時停止と再開

仮想マシンの一時停止や再開を行いたい場合は、下記のようなコマンドを使用します:

`stop`

仮想マシンの動作を一時的に停止します。

`cont`

一時停止していた仮想マシンを再開します。

`system_reset`

仮想マシンをリセットします。物理マシンでリセットボタンを押した場合と同じ動作になります。これにより、ファイルシステムが不安定な状態になる可能性があります。

`system_powerdown`
: マシンに対して ACPI のシャットダウン要求を送信します。物理マシンで電源ボタンを押した場合と同じ動作になります。

`q` または `quit`
: QEMU を即時に終了します。

37.10 ライブマイグレーション

ライブマイグレーションを行うことで、一方のホストシステムから他方のホストシステムに対して、仮想マシンを動作させた状態のまま移動することができます。恒久的にホストを移動することができるほか、メンテナンスなどで一時的に移動することもできます。

ライブマイグレーションを行う際の要件は下記のとおりです:

- 16.2項「移行における要件」に示されている全ての要件が満たされていること。
- 移行元と移行先の VM ホストサーバ で、同じ CPU 機能を有していること。
- AHCI インターフェイスや VirtFS 機能、`-mem-path` コマンドラインオプションを指定していないこと (いずれもライブマイグレーションとは互換性がありません) 。
- 移行元と移行先のホストが同じ方法で起動していること。
- QEMU のコマンドラインオプション `-snapshot` は移行に際して使用すべきではありません (サポート対象外でもあります) 。

重要: サポート状態について

openSUSE Leap では `postcopy` モードはサポートしていません。これは技術プレビューとしてのみ提供されているものです。`postcopy` モードに関する詳細は、https://wiki.qemu.org/Features/PostCopyLiveMigration (英語) をお読みください。

また、さらに詳しい推奨条件が https://www.linux-kvm.org/page/Migration に書かれています。

ライブマイグレーションは下記の手順で行います:

1. 移行元のホストで仮想マシンのインスタンスが動作していることを確認します。

2. 移行先のホストで、仮想マシンを frozen listening (凍結待ち受け) モードで起動します。具体的には移行元のホストのコマンドラインパラメータに加えて、`-incoming tcp:IP:ポート` のパラメータを追加します。ここで、`IP` には IP アドレスを、`ポート` には移行を待ち受けるポートをそれぞれ指定します。なお、IP アドレスに 0 を指定した場合、全てのインターフェイスで待ち受けることになります。

3. 移行元のホストでモニタコンソールを表示させ、`migrate -d tcp:移行先_IP:ポート` のように入力して実行し、ライブマイグレーションを開始します。

4. ライブマイグレーションの状態を確認したい場合は、移行元のホストのモニタコンソールで `info migrate` コマンドを実行します。

5. ライブマイグレーションをキャンセルするには、移行元のホストのモニタコンソールで `migrate_cancel` コマンドを実行します。

6. ライブマイグレーションで許容できる最大限のダウンタイムを秒単位で指定したい場合は、`migrate_set_downtime 秒数` のように入力して実行します。

7. ライブマイグレーションの最大速度をバイト毎秒単位で指定したい場合は、`migrate_set_speed バイト毎秒` のように入力して実行します。

37.11 QMP - QEMU マシンプロトコル

QMP は JSON ベースのプロトコルで、`libvirt` のようなアプリケーションと動作中の QEMU インスタンスの間で通信を行うことができるプロトコルです。QMP プロトコルを使用することで、QEMU モニタにさまざまな方法でアクセスすることができるようになります。

37.11.1 標準入出力経由での QMP アクセス

QMP を使用する際の最も柔軟な方法は、`-mon` オプションを指定する方法です。下記の例では、標準入出力を利用して QMP のインスタンスを作成しています。ただし、下記の例では `->` がクライアントから QEMU のインスタンス宛のデータを、`<-` が QEMU から返された出力をそれぞれ表しています。

```
> sudo qemu-system-x86_64 [...] \
-chardev stdio,id=mon0 \
-mon chardev=mon0,mode=control,pretty=on

<- {
```

```
    "QMP": {
        "version": {
            "qemu": {
                "micro": 0,
                "minor": 0,
                "major": 2
            },
            "package": ""
        },
        "capabilities": [
        ]
    }
}
```

QMP の接続が確立すると、QMP は "ようこそ" メッセージを送信し、機能ネゴシエーションモードに移行します。このモードでは、`qmp_capabilities` コマンドのみを動作させることができます。機能ネゴシエーションモードを終了して通常のコマンドモードに移行したい場合は、まず `qmp_capabilities` コマンドを送信しなければなりません:

```
-> { "execute": "qmp_capabilities" }
<- {
    "return": {
    }
}
```

なお、`"return": {}` は QMP の成功応答を意味します。

QMP ではコマンドにパラメータを指定することができます。たとえば CD-ROM ドライブのメディアを取り出したい場合は、下記のように入力して送信します:

```
->{ "execute": "eject", "arguments": { "device": "ide1-cd0" } }
<- {
    "timestamp": {
        "seconds": 1410353381,
        "microseconds": 763480
    },
    "event": "DEVICE_TRAY_MOVED",
    "data": {
        "device": "ide1-cd0",
        "tray-open": true
    }
}
{
    "return": {
    }
}
```

37.11.2 Telnet 経由での QMP アクセス

標準入出力を使用する代わりに、QMP インターフェイスをネットワークソケットに接続して使用することもできます:

```
> sudo qemu-system-x86_64 [...] \
-chardev socket,id=mon0,host=localhost,port=4444,server,nowait \
-mon chardev=mon0,mode=control,pretty=on
```

あとは telnet を起動してポート 4444 に接続します:

```
> telnet localhost 4444
Trying ::1...
Connected to localhost.
Escape character is '^]'.
<- {
    "QMP": {
        "version": {
            "qemu": {
                "micro": 0,
                "minor": 0,
                "major": 2
            },
            "package": ""
        },
        "capabilities": [
        ]
    }
}
```

必要であれば、複数のモニタインターフェイスを同時に作成することもできます。下記の例では、「通常の」QEMU モニタコマンドを解釈する HMP インスタンスを標準入出力に作成し、追加でローカルホストのポート 4444 に QMP インスタンスを作成しています:

```
> sudo qemu-system-x86_64 [...] \
-chardev stdio,id=mon0 -mon chardev=mon0,mode=readline \
-chardev socket,id=mon1,host=localhost,port=4444,server,nowait \
  -mon chardev=mon1,mode=control,pretty=on
```

37.11.3 Unix ソケット経由での QMP アクセス

QEMU を起動する際に `-qmp` オプションを指定することで、Unix ソケットを作成することができます:

```
> sudo qemu-system-x86_64 [...] \
-qmp unix:/tmp/qmp-sock,server --monitor stdio
```

```
QEMU waiting for connection on: unix:./qmp-sock,server
```

上記の例で作成した `/tmp/qmp-sock` を介して QEMU インスタンスと通信を行うには、同じホスト内でもう 1 つの端末を開いて、`nc` コマンド (詳しくは `man 1 nc` をお読みください) を使用します:

```
> sudo nc -U /tmp/qmp-sock
<- {"QMP": {"version": {"qemu": {"micro": 0, "minor": 0, "major": 2} [...]
```

37.11.4 `libvirt` の `virsh` コマンド経由での QMP アクセス

`libvirt` (詳しくは パートII「libvirt を利用した仮想マシンの管理」をお読みください) 内で仮想マシンを動作させている場合は、`virsh qemu-monitor-command` を実行することで、動作中のゲストと通信を行うことができます:

```
> sudo virsh qemu-monitor-command vm_guest1 \
--pretty '{"execute":"query-kvm"}'
<- {
    "return": {
        "enabled": true,
        "present": true
    },
    "id": "libvirt-8"
}
```

上記の例では、ホスト側に KVM を動作させる機能が存在しているかどうかと、KVM が有効化されているかどうかを調べる、シンプルな `query-kvm` コマンドを実行しています。



ヒント: 読みやすい出力の生成方法について

JSON 形式ではなく分かりやすい QEMU 出力を使用したい場合は、`--hmp` オプションを指定して実行してください:

```
> sudo virsh qemu-monitor-command vm_guest1 --hmp "query-kvm"
```

VI トラブルシューティング

38 内蔵ヘルプとパッケージのドキュメンテーション 375

39 システム情報とログの収集 377

38 内蔵ヘルプとパッケージのドキュメンテーション

改訂履歴

2024-06-27

様々な仮想化向けパッケージには、仮想化ホストを様々な観点から管理するためのコマンドが用意されています。これらのコマンドについて全てのオプションを覚えるのは無理ですし、現実的でもありません。また Xen や KVM のホストの基本パッケージには、シェルコマンドに対するマニュアルページと組み込みヘルプが用意されています。このほか、ドキュメンテーションのサブパッケージをインストールすることで、基本的なインストールには含まれていない、追加のドキュメントをインストールすることもできます。

シェルコマンドに対するマニュアルページ

ほとんどのコマンドには、そのコマンドやオプションの使用方法、そしてコマンドの実行例までもが書かれた詳しいマニュアルページ (多くは英語のみ) が同梱されています。たとえば `virt-install` コマンドに対するマニュアルページを表示したい場合は、下記のようなコマンドを実行します:

```
> man virt-install
```

シェルコマンドの内蔵ヘルプ

コマンド内には、簡潔な説明が分野別に並べられた内蔵ヘルプ (多くは英語のみ) も用意されています。たとえば `virt-install` コマンドに対する簡潔な説明を読みたい場合は、下記のようなコマンドを実行します:

```
> virt-install --help
```

内蔵ヘルプには特定のオプションに対してさらに詳しい説明 (多くは英語のみ) が用意されている場合もあります。たとえばディスク関連のオプションに対するサブオプションを一覧表示したい場合は、下記のようなコマンドを実行します:

```
> virt-install --disk help
```

ドキュメンテーションのサブパッケージ

多くの仮想化パッケージには、ドキュメンテーションサブパッケージの形式で追加のコンテンツ (多くは英語のみ) が提供されています。たとえば `libvirt-doc` パッケージには、https://libvirt.org で提供されている全てのドキュメンテーションが含まれているほか、libvirt C 言語 API の使用方法を説明するためのサンプルコードなども含まれています。なお、ドキュメンテー

ションサブパッケージ内に含まれているファイルを一覧表示したい場合は、`rpm` コマンドをお使いください。たとえば `libvirt-doc` パッケージに含まれているファイルの一覧を表示するには、下記のようなコマンドを実行します:

```
rpm -ql libvirt-doc
```

39 システム情報とログの収集

改訂履歴

2023-07-17

なお、場合によっては 問題を解決するには、独自のログやデバッグ設定を行って調査する必要があるかもしれません。

39.1 libvirt のログ制御

`libvirt` ではライブラリとデーモンの両方に対してログを設定することができます。ログ機構の動作はログレベルとフィルタ、そして出力設定で調整することができます。

ログレベル

`libvirt` のログメッセージには 4 種類のうちのいずれかの優先度レベルが設定されています。それぞれ DEBUG (デバッグ), INFO (情報), WARNING (警告), ERROR (エラー) と呼ばれ、DEBUG は短時間に数ギガバイトにも及ぶような最も多く非常に詳細な (冗長な) 出力を表し、後は多い順に INFO, WARNING, ERROR となります。なお、既定のログレベルは ERROR です。

ログフィルタ

ログフィルタは、特定のコンポーネントとログレベルのみに限定して出力を行うための方法です。ログフィルタを設定することで、特定のコンポーネントのみ DEBUG で出力し、それ以外を ERROR のみに設定することもできます。既定ではログフィルタの設定は行われていません。

ログ出力先

ログ出力先は、フィルタ設定したログメッセージの送信先を指定するための仕組みです。ログメッセージはファイルに出力できるほか、プロセスの標準エラー出力や journald などを指定することができます。既定ではフィルタ設定したログメッセージは journald に送信されます。

`libvirt` のログ出力について、詳しくは https://libvirt.org/logging.html をお読みください。

`libvirt` の既定のインストール状態では、ログレベルは ERROR に設定され、ログフィルタは何も設定されません。また、ログ出力は journald になります。この場合 `libvirt` デーモンからのログメッセージは、`journalctl` コマンドで表示することができます:

```
# journalctl --unit libvirtd
```

なお、通常の運用範囲においては既定のログ設定で十分であり、`libvirt` のアプリケーションやユーザに対して適切な範囲でメッセージを送信しますが、内部の問題に直面した場合は、DEBUG レベルのメッセージが必要となることがしばしばあります。たとえば `libvirt` と QEMU モニタの間でのやり

取りに問題があると考える場合などがそれにあたります。このような場合は、`libvirt` と QEMU の通信のみに絞ってデバッグメッセージを出力させてください。たとえば下記の例では、QEMU ドライバが出力するデバッグメッセージを、`/tmp/libvirtd.log` というファイルに出力しています。

```
log_filters="1:qemu.qemu_monitor_json"
log_outputs="1:file:/tmp/libvirtd.log"
```

`libvirt` デーモンのログ制御は `/etc/libvirt/libvirtd.conf` で行います。なお、設定変更後はデーモンを再起動しなければならないことに注意してください。

```
# systemctl restart libvirtd.service
```

用語集

一般

Dom0

Xen 環境で使用される用語で、仮想マシンを意味します。ホスト側のオペレーティングシステムは実際には仮想マシンで、特権ドメイン下で動作するものであるため、これを Dom0 と呼んでいます。ホスト内でのその他の仮想マシンは非特権ドメイン下で動作するものであることから、これらは DomU と呼びます。

KVM

第4章「KVM 仮想化の紹介」をお読みください。

VHS

仮想ホストサーバ (Virtualization Host Server) の略です。

SUSE による仮想化プラットフォームソフトウエアを動作させている物理的なコンピュータを意味します。仮想環境にはハイパーバイザとホスト環境、仮想マシンと関連ツール、コマンド、設定ファイルがそれぞれ含まれています。一般には、単純に ホストやホストコンピュータ、ホストマシン (HM)、仮想サーバ (VS)、仮想マシンホスト (VMH)、VM ホストサーバ (VHS) などと呼ぶ場合もあります。

VirtFS

VirtFS は新しい準仮想化ファイルシステムインターフェイスで、KVM 環境でパススルー技術を改善するために作られた仕組みです。VirtIO フレームワークをベースにしています。

Xen

第3章「Xen 仮想化の紹介」をお読みください。

xl

Xen 向けのコマンド集で、管理者がホストコンピュータのコマンドラインを使用することで、仮想マシンの管理を行うことができるものです。廃止予定である `xm` ツールスタックの置き換えとして作られているものです。

ハイパーバイザ

仮想マシンと物理的なコンピュータハードウエアとの間を、低レベルな (ハードウエアに近い) 範囲で仲介するソフトウエアを意味します。

ハードウエア支援
: Intel* と AMD* は、いずれも仮想に対するハードウエア支援技術を提供しています。これらの技術により、ソフトウエア側が主なオーバーヘッドとなっていた仮想化で、VM の入出力処理の頻度 (VM トラップ) を減らし、ハードウエアで実行することで、効率を上げることができます。これに加えて、メモリのフットプリントも減らすことができますので、リソース制御もより効率的に行うことができるほか、特定の I/O デバイスの割り当てもより安全に行うことができるようになっています。

ホスト環境
: ホストコンピュータの環境とやり取りを行うことのできる、デスクトップもしくはコマンドライン環境を意味します。コマンドライン環境のほか、GNOME や IceWM などのグラフィカルなデスクトップ環境を使用することもできます。ホスト環境は仮想マシンの特殊形態を実行する仕組みで、仮想マシンの制御や管理などの権限が与えられています。一般的には、Dom0 や特権ドメイン、ホストオペレーティングシステムなどと呼ぶこともあります。

仮想マシン
: ゲスト側のオペレーティングシステムや対応するアプリケーションを動作させることのできる、仮想化された PC 環境 (VM) を意味します。VM ゲスト と呼ばれることもあります。

仮想マシンの作成ウイザード
: YaST 内と仮想マシンマネージャ内に用意されているソフトウエアプログラムで、仮想マシンを作成するための手順を示すグラフィカルなインターフェイスを提供する仕組みです。テキストモードで仮想マシンを作成したい場合は、`virt-install` コマンドを利用します。

仮想マシンマネージャ
: 仮想マシンを作成したり管理したりするためのグラフィカルなインターフェイスを提供する、ソフトウエアプログラムを意味します。

仮想化ゲスト
: 仮想マシン内で動作するオペレーティングシステムやアプリケーションを意味します。

準仮想化フレームバッファ
: 準仮想化モードで仮想マシン内の表示に関わるフレームデータを含む、メモリバッファ内のビデオディスプレイの出力デバイスです。

CPU

CPU オーバーコミット
: 仮想 CPU のオーバーコミットとは、物理システム内に存在する CPU 数より多くの仮想 CPU を、仮想マシンに割り当てることのできる機能を意味します。この機能はシステムの全体性能を強化する目的では利用できませんが、テスト目的には有用です。

CPU キャッピング
: CPU キャッピングの機能は、物理的な CPU 性能を仮想マシンに提供する際に、割合を指定して制限を行うための機能を意味します。

CPU ピニング
: CPU ピニングはプロセッサアフィニティとも呼ばれ、特定のプロセスやスレッドに対して、特定の 1 台もしくは複数台の中央処理装置 (CPU) を使用するようにする仕組みを意味します。

CPU ホットプラグ
: CPU ホットプラグとは、システムのシャットダウンを行うことなく、CPU の置換や追加、抜去などを行う機能を意味します。

ネットワーク

ネットワークアドレス変換 (Network Address Translation (NAT))
: 仮想マシンがホストの IP アドレスや MAC アドレスを使用するタイプのネットワーク接続を意味します。

ブリッジ型ネットワーク
: 仮想マシンを外部ネットワークに存在するものとして切り出すタイプのネットワーク接続で、ホストコンピュータとは切り離され、無関係な存在として位置づけられるものを意味します。

ホスト無しブリッジ
: 物理的なネットワークデバイスが存在するものの、ホスト側の仮想ネットワークデバイスを持たないタイプのネットワークブリッジを意味します。これにより、仮想マシンは外部ネットワークとの通信を行うことができるものの、ホストとの間は通信を行うことができなくなります。これにより、ホスト環境からは独立した仮想的なネットワーク通信を実現することができます。

ローカルブリッジ
: 仮想ネットワークデバイスが存在するものの、ホスト側の物理的なネットワークデバイスを持たないタイプのネットワークブリッジを意味します。これにより、仮想マシンはホストとの間のほか、ホスト内

の他の仮想マシンとも通信を行うことができます。仮想マシンはホストを介して外部と通信することになります。

内部ネットワーク
: 仮想マシンの通信相手がホスト環境に限定されたネットワーク設定を意味します。

外部ネットワーク
: ホストが接続されているローカルネットワーク環境の外側のネットワークを意味します。

従来型ブリッジ
: ホスト側が提供する物理的なネットワークデバイスと、仮想ネットワークデバイスの両方を持つタイプのネットワークブリッジを意味します。

空ブリッジ
: ホスト側が提供する物理的なネットワークデバイスや、仮想ネットワークデバイスを持たないタイプのネットワークブリッジで、これにより他の仮想マシンとの通信はできるものの、ホストや外部ネットワークとは通信のできない環境になるものを意味します。

ストレージ

AHCI
: AHCI は Advanced Host Controller Interface の略で、Intel* 社が規定したシリアル ATA (SATA) ホストバスアダプタの仕様を意味します。なお、実装固有の箇所は含まない構造です。

xvda
: 準仮想化マシンにおける 1 台目の仮想ディスクを意味する名前です。

スパースイメージファイル
: ファイルシステム内に指定されたサイズをそのまま予約するのではなく、書き込み時に必要に応じてサイズを拡張するタイプのディスクイメージファイルを意味します。

ファイルベースの仮想ディスク
: ディスクイメージファイルとも呼ばれる、ファイルベースの仮想ディスクです。

ブロックデバイス
: CD-ROM ドライブやディスクドライブなど、データをブロックとして管理するタイプのデータストレージデバイスを意味します。パーティションやボリュームなどもブロックデバイスと考えられます。

純粋ディスク

ファイルシステムを介することなく、バイト単位でディスクに直接アクセスするタイプのディスクを意味します。

略語

ACPI

Advanced Configuration and Power Interface (ACPI) の略で、オペレーティングシステムからのデバイスの設定や電源管理のための統一規格を意味します。

AER

Advanced Error Reporting

AER は PCI Express 仕様で提供されている機能のうちの 1 つで、PCI でのエラーを報告し、それらのうちのいくつかを修復することのできる仕組みを提供します。

APIC

Advanced Programmable Interrupt Controller (APIC) は割り込みコントローラの一種です。

BDF

Bus (バス):Device (デバイス):Function (機能)

PCI や PCIe のデバイスを簡潔に表現するための表記法です。

CG

Control Groups (コントロールグループ)

リソース (CPU, メモリ, ディスク I/O など) を制限したり測定したり、孤立させたりするための機能を指します。

EDF

Earliest Deadline First

このスケジューラは直感的な重み付き CPU 共有の仕組みで、時間面の保証を行うためのリアルタイムアルゴリズムを意味します。

EPT

Extended Page Tables (拡張ページテーブル)

仮想環境での性能はネイティブな (仮想化を行わない) 環境に近くなっていますが、仮想化によって少しのオーバーヘッドが存在しています。オーバーヘッドは CPU の仮想化機能 (MMU) のほか、I/O デバイスによっても発生します。最近の x86 プロセッサでは、AMD, Intel ともにハードウエア拡張を提供し、このような性能ギャップを埋める仕組みを用意しています。2006 年、AMD は

AMD-Virtualization (AMD-V) 技術、Intel は Intel® VT-x 技術を発表し、仮想化に対する初めてのハードウエアサポートを提供するようになりました。最近になって Intel は、MMU の仮想化を行う第 2 世代の仮想化ハードウエアサポートを提供するようになりました。これを Extended Page Tables (拡張ページテーブル; EPT) と呼んでいます。EPT が有効化されたシステムの場合、MMU の仮想化にあたって、シャドウページを使用するのに比べて性能を改善することができるようになっています。EPT では、負荷状況にもよりますが、メモリアクセスの遅延が発生します。ただし、ゲストとハイパーバイザでラージページを使用することによって、そのような遅延を効率的に小さくすることができるようになっています。

FLASK

Flux Advanced Security Kernel

Xen は FLASK と呼ばれるセキュリティアーキテクチャを採用していて、同名のモジュールによる強制アクセス制御を実装しています。

HAP

High Assurance Platform

HAP はハードウエア技術とソフトウエア技術の組み合わせで、ワークステーションやネットワークのセキュリティを改善するための仕組みです。

HVM

Hardware Virtual Machine (ハードウエア仮想マシン) の略です (Xen ではこのように呼ばれています) 。

IOMMU

Input/Output Memory Management Unit

IOMMU (AMD* の技術) はメモリ管理ユニット (MMU) を意味する技術で、ダイレクトメモリアクセス (DMA) に対応した I/O バスをメインメモリに接続することができるものを指します。

KSM

Kernel Same Page Merging

KSM は、ゲストとホストで同一の内容を持つメモリページを自動的に共有させる仕組みで、VM ホストサーバ 側で KSM が有効化されていれば、KVM は KSM を最適に使用することができます。

MMU

Memory Management Unit (メモリ管理ユニット)

CPU 側から要求されたメモリへのアクセスを扱うハードウエアコンポーネントを意味する用語です。仮想アドレスから物理アドレスへの翻訳を行う (仮想メモリ管理と呼びます) ほか、メモリの保護やキャッシュ制御、バス調停などを提供します。古いコンピュータアーキテクチャ (特に 8 ビットシステム) では、バンク切り替えなどと呼ばれていたものです。

PAE

Physical Address Extension (物理アドレス拡張)

32 ビットの x86 オペレーティングシステムでは、Physical Address Extension (PAE) を利用することで、4 GB 以上の物理メモリにアクセスすることができるようになっています。PAE モードでは、ページテーブルエントリ (PTE) は 64 ビット幅になります。

PCID

Process-context identifiers (プロセス-コンテキスト識別子)

これらは、論理プロセッサが複数の線形アドレス空間の情報をキャッシュする機能を意味し、ソフトウエアが異なる線形アドレス空間に切り替えた際にも、キャッシュ情報を維持することができるようにするものです。INVPCID 命令はきめ細かい TLB (TLB) フラッシュに使用するもので、カーネルにとって有益な仕組みです。

PCIe

Peripheral Component Interconnect Express

PCIe は古い規格である PCI, PCI-X, AGP バス標準を置き換えるために設計された仕組みです。PCIe は従来よりも高い最大システムバススループットや少ない I/O ピン数、そしてより小さくなった物理フットプリントなど、さまざまな改善が加えられています。これに加えて、より詳細なエラー検出・レポート機構 (AER) が用意され、ホットプラグにもネイティブ対応するようになっています。なお、PCI との後方互換性も維持されています。

PSE, PSE36

Page Size Extended (ページサイズ拡張)

PSE は x86 プロセッサの機能で、従来の 4 KiB サイズよりも大きなページサイズに対応するための仕組みです。PSE-36 は通常の 10 ビットに加えて 4 ビット分の拡張が行われ、ラージページを指し示すページディレクトリエントリ内で使用されています。これにより、36 ビットのアドレス空間内でラージページを使用することができるようになっています。

PT

Page Table (ページテーブル)

ページテーブルは、コンピュータのオペレーティングシステム内にある仮想メモリシステムが使用するデータ構造で、仮想アドレス (プロセスごとに個別管理されているアドレス) と物理アドレス (実際のハードウエアアドレス) の対応付けを管理するための仕組みです。

QXL

QXL は仮想環境向けの Cirrus VGA フレームバッファ (8M) です。

RVI / NPT

Rapid Virtualization Indexing, Nested Page Tables

AMD の第 2 世代仮想化ハードウエア支援技術で、プロセッサのメモリ管理ユニット (MMU) 向けの仕組みです。

SATA

Serial ATA (シリアル ATA)

SATA はコンピュータバスインターフェイスの一種で、ホストバスアダプタからハードディスクや光学ドライブなど、ストレージデバイスなどに接続するためのインターフェイスです。

Seccomp2 ベースのサンドボックス

不正な動作を防ぐための追加保護として、特定のシステムコールのみを許可するようなサンドボックス環境を意味します。

SMEP

Supervisor Mode Execution Protection

Xen ハイパーバイザで提供される仕組みで、ユーザモードページの実行を阻止するための仕組みです。これにより、アプリケーションからハイパーバイザへの攻撃の多くを困難にします。

SPICE

Simple Protocol for Independent Computing Environments の略です。

SXP

SXP ファイルは Xen の設定ファイルを意味します。

TCG

Tiny Code Generator

インストラクションを CPU で直接実行するのではなく、擬似的に実行することを意味します。

THP

Transparent Huge Pages (透過型ヒュージページ)

CPU が既定の 4KB より大きなページを利用してメモリにアクセスする機能を意味します。これにより、メモリの使用率と CPU キャッシュの使用率を削減することができます。VM ホストサーバ 側で THP が有効化されていれば、KVM は THP を (madvise と日和見的な方式で) 最適に使用することができます。

TLB

Translation Lookaside Buffer

TLB はメモリ管理ハードウエアが仮想アドレスへの変換速度を上げるためのキャッシュです。現在ではデスクトップやノート PC 、サーバのいずれのプロセッサでも、TLB を利用して仮想アドレスから物理アドレス空間への変換を行っています。また、仮想メモリを使用するほぼ全てのハードウエア内に搭載されている仕組みでもあります。

VCPU

仮想化された CPU の状態を含む、スケジュール対象の実体を意味します。

VDI

Virtual Desktop Infrastructure (仮想デスクトップインフラストラクチャ)

VFIO

カーネルバージョン 3.6 およびそれ以降で導入された仕組みで、ユーザスペースから PCI デバイスにアクセスするための新しい方式を提供するものです。

VHS

Virtualization Host Server (仮想化ホストサーバ)

VM ルート

VMM は VMX ルート操作内で動作し、ゲスト側のソフトウエアは VMX 非ルート操作内で動作します。VMX ルート操作と VMX 非ルート操作との間の遷移は、VMX 遷移と呼ばれます。

VMCS

Virtual Machine Control Structure (仮想マシン制御構造)

VMX の非ルート操作と VMX の遷移は Virtual Machine Control Structure (仮想マシン制御構造) (VMCS) と呼ばれるデータ構造によって制御されます。VMCS へのアクセスは VMCS ポインタ (論理プロセッサごとに 1 つ) と呼ばれるプロセッサの状態コンポーネントを介して管理されます。VMCS ポインタの値は VMCS に対する 64 ビットのアドレスです。VMCS ポインタは VMPTRST や VMPTRLD のインストラクションを利用することで、読み込みおよび書き込みを行うことができます。VMM は VMREAD, VMWRITE, VMCLEAR の各インストラクションを利用して、VMCS の設定を行います。VMM では対応するそれぞれの仮想マシンに対して、異なる VMCS を使用することができます。複数の論理プロセッサ (仮想プロセッサ) を持つ仮想マシンの場合、VMM はそれぞれの仮想プロセッサに対して異なる VMCS を使用することができます。

VMDq

Virtual Machine Device Queue (仮想マシンデバイスキュー)

ハードウエアレベルで複数の VM に対応できるマルチキュー型のネットワークアダプタにより、各パケットキューをそれぞれの VM (VM の IP アドレス) に割り当てることができるようになります。

VMM

Virtual Machine Monitor (Hypervisor) (仮想マシンモニタ (ハイパーバイザ))

プロセッサがハイパーバイザ (VMM) に関係するインストラクションやイベントを検出すると、ゲストモードを終了して VMM に戻ることができます。VMM はネイティブとほとんど変わらない速度でインストラクションやその他のイベントを擬似し、ゲストモードに戻ることができます。ゲストモードからの VMM への遷移、および VMM からゲストモードへの遷移は、ゲスト側の実行が完全に停止することになるため、大きな遅延のある操作となります。

VMX

Virtual Machine eXtensions (仮想マシン拡張)

VPID

TLB のソフトウエア制御に対する新しいサポートです (小さな VMM の開発努力によって、 TLB の性能を改善するためのものです) 。

VT-d

Virtualization Technology for Directed I/O

Intel* (https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices) 向けの IOMMU です。

vTPM

Trusted Computing を介して、ゲスト向けのセキュリティ機能を提供するコンポーネントです。

A NVIDIA カードに対する GPU パススルー の設定

改訂履歴

2023-12-22

A.1 概要

本章では、ホストマシンに搭載された NVIDIA GPU グラフィックカードを仮想化ゲストで使用するための手順を説明しています。

A.2 事前要件

- GPU パススルーは AMD64/Intel 64 アーキテクチャでのみサポートされています。
- 本章で説明している内容は、V100/T1000 NVIDIA ベースのカードを使用することを前提にしています。つまり、GPU を計算処理の目的でのみ使用する想定です。
- お使いの製品が NVIDIA Tesla 製品であることを確認しておいてください。具体的には Maxwell, Pascal, Volta の各アーキテクチャである必要があります。
- 本章内の設定を実施すると、対象の GPU はホスト側からは使用できなくなります。そのため、表示用の別のグラフィックカードを接続しておくか、もしくは SSH でリモートアクセスできることをあらかじめ確認しておいてください。

A.3 ホスト側の設定

A.3.1 ホスト側の環境確認

1. また、お使いのホストが VT-d に対応していて、かつファームウエア側の設定で有効化されていることを確認します:

```
> dmesg | grep -e "Directed I/O"
[   12.819760] DMAR: Intel(R) Virtualization Technology for Directed I/O
```

ファームウエア側の設定で VT-d が有効化されていない場合は、ホストを再起動して有効化してください。

2. また、表示用の追加 GPU もしくは VGA カードが搭載されていることを確認します:

```
> lspci | grep -i "vga"
07:00.0 VGA compatible controller: Matrox Electronics Systems Ltd. \
  MGA G200e [Pilot] ServerEngines (SEP1) (rev 05)
```

Tesla V100 カードを使用する場合は、下記のようにしてカードが認識されていることを確認します:

```
> lspci | grep -i nvidia
03:00.0 3D controller: NVIDIA Corporation GV100 [Tesla V100 PCIe] (rev a1)
```

T1000 Mobile カードを使用する場合 (たとえば Dell 5540 など) は、下記のようにしてカードが認識されていることを確認します:

```
> lspci | grep -i nvidia
01:00.0 3D controller: NVIDIA Corporation TU117GLM [Quadro T1000 Mobile] (rev a1)
```

A.3.2 IOMMU の有効化

既定では IOMMU は無効化されています。有効化するには、`/etc/default/grub` ファイルを編集して、起動時に有効化しておく必要があります。

1. Intel プロセッサの場合は下記の内容を追加します:

```
GRUB_CMDLINE_LINUX="intel_iommu=on iommu=pt rd.driver.pre=vfio-pci"
```

AMD プロセッサの場合は下記の内容を追加します:

```
GRUB_CMDLINE_LINUX="iommu=pt amd_iommu=on rd.driver.pre=vfio-pci"
```

2. `/etc/default/grub` ファイルを編集して保存したら、あとはメインの GRUB 2 設定ファイルである `/boot/grub2/grub.cfg` ファイルを再生成します:

```
> sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. ホストを再起動して、IOMMU が有効化されていることを確認します:

```
> dmesg |  grep -e DMAR -e IOMMU
```

A.3.3 Nouveau ドライバのブラックリスト設定

NVIDIA カードを VM ゲストに割り当てるには、ホスト側の OS で NVIDIA GPU 向けの内蔵ドライバである `nouveau` を読み込まないように設定する必要があります。具体的には `/etc/modprobe.d/60-blacklist-nouveau.conf` ファイルを作成して下記のような内容を記述します:

```
blacklist nouveau
```

A.3.4 VFIO の設定とパススルーのための GPU 分離

1. あとはパススルー設定を行うカードの製造元と型番 ID を調べます。この場合、A.3.1項「ホスト側の環境確認」で調べたバス番号を指定して表示させます。たとえば `03:00.0` の場合、下記のようになります:

   ```
   > lspci -nn | grep 03:00.0
   03:00.0 3D controller [0302]: NVIDIA Corporation GV100 [Tesla V100 PCIe] [10de:1db4]
    (rev a1)
   ```

2. `/etc/modprobe.d/vfio.conf` ファイルを作成して、下記のような内容を記述します:

   ```
   options vfio-pci ids=10de:1db4
   ```

   注記

   また、お使いのカード側で追加の `ids=` を指定する必要があるかどうかを確認してください。これは、カードによってはオーディオデバイスも合わせて指定しなければ、カードを使用できなくなってしまうものがあるためです。

A.3.5 VFIO ドライバの読み込み

VFIO ドライバの読み込みにあたっては、下記の 3 種類の方法が用意されています。

A.3.5.1 initrd ファイル内へのドライバの組み込み

1. `/etc/dracut.conf.d/gpu-passthrough.conf` ファイルを作成して、下記のような内容を記述します (頭に空白を入れる必要があることに注意してください):

   ```
   add_drivers+=" vfio vfio_iommu_type1 vfio_pci vfio_virqfd"
   ```

2. あとは initrd ファイルを再作成します:

```
> sudo dracut --force /boot/initrd $(uname -r)
```

A.3.5.2 ドライバの自動読み込み設定

`/etc/modules-load.d/vfio-pci.conf` ファイルを作成して、下記のような内容を記述します:

```
vfio
vfio_iommu_type1
vfio_pci
kvm
kvm_intel
```

A.3.5.3 ドライバの手動読み込み

システム稼働中にドライバを手作業で読み込みたい場合は、下記のようなコマンドを実行します:

```
> sudo modprobe vfio-pci
```

A.3.6 Microsoft Windows ゲストに対する MSR の無効化

Microsoft Windows ゲストを使用する場合、ゲスト側のクラッシュを防ぐため、MSR (model-specific register) の無効化をお勧めします。無効化を行うには、`/etc/modprobe.d/kvm.conf` ファイルを作成して、下記のような内容を記述します:

```
options kvm ignore_msrs=1
```

A.3.7 UEFI ファームウエアのインストール

GPU パススルーを正しく動作させるためには、ホスト側を UEFI ファームウエアで起動する必要があります (つまり、従来の BIOS 起動手順を使用しないようにする必要があります) 。また、`qemu-ovmf` パッケージをインストールしていない場合は、まずインストールを行ってください:

```
> sudo zypper install qemu-ovmf
```

A.3.8 ホストマシンの再起動

これまでの設定変更を有効化するため、ここでホスト側のマシンを再起動してください:

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

A.4 ゲスト側の設定

本章では、ゲスト側の仮想マシンでホスト側の NVIDIA GPU を使用するための設定方法を説明しています。ゲスト側の仮想マシンをインストールするには、仮想マシンマネージャ もしくは `virt-install` をお使いください。詳しくは 第9章「ゲストのインストール」をお読みください。

A.4.1 ゲスト側の設定を行うための要件

ゲスト側の仮想マシンをインストールする際、［インストールの前に設定をカスタマイズする］を選択して、下記のとおりデバイスの設定を行ってください:

- 可能であれば Q35 チップセットをお使いください。
- UEFI ファームウエアを利用してゲスト VM をインストールしてください。
- 下記の擬似デバイスを追加しておいてください:
  グラフィック: Spice もしくは VNC
  デバイス: qxl, VGA, Virtio のいずれか
  詳しくは 13.6項「ビデオ」をお読みください。
- ゲスト側にホスト側の PCI デバイス (ここまでの例では `03:00.0` になっています) を追加します。詳しくは 13.12項「VM ゲスト に対するホスト側の PCI デバイスの割り当て」をお読みください。
- 最適な性能を引き出すため、ネットワークカードとストレージに対して virtio ドライバを使用しておくことをお勧めします。

A.4.2 グラフィックカードドライバのインストール

A.4.2.1 Linux ゲストの場合

手順 A.1: RPM ベースのディストリビューションの場合

1. https://www.nvidia.com/download/driverResults.aspx/131159/en-us から RPM パッケージをダウンロードします。
2. ダウンロードした RPM パッケージをインストールします:

```
> sudo rpm -i nvidia-diag-driver-local-repo-sles123-390.30-1.0-1.x86_64.rpm
```

3. リポジトリを更新して `cuda-drivers` パッケージをインストールします。下記のコマンドは SUSE ディストリビューション向けですので、それ以外のものをお使いの場合はそれぞれに合わせて実施してください:

```
> sudo zypper refresh && zypper install cuda-drivers
```

4. ゲスト VM を再起動します:

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

手順 A.2: 一般的なインストーラを使用する場合

1. 一般的なインストーラを使用する場合、NVIDIA ドライバモジュールをコンパイルする必要があることから、`gcc-c++` と `kernel-devel` の各パッケージをインストールしておいてください。
2. また、NVIDIA が提供するドライバには署名が付与されていないことから、ゲスト側で Secure Boot を使用している場合は、無効化してください。YaST GRUB 2 モジュールを使用することで、Secure Boot を無効化することができます。詳しくは『リファレンス』、第14章「UEFI (Unified Extensible Firmware Interface)」、14.1.1項「openSUSE Leap での実装」をお読みください。
3. あとは https://www.nvidia.com/Download/index.aspx?lang=ja-jp からドライバのインストールスクリプトをダウンロードし、実行可能な状態にしてから実行してください。これでドライバのインストールが完了します:

```
> chmod +x NVIDIA-Linux-x86_64-460.73.01.run
> sudo ./NVIDIA-Linux-x86_64-460.73.01.run
```

4. あとは CUDA ドライバを https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=SLES&target_version=15&target_type=rpmlo からダウンロードして実行し、画面上に表示された手順に従ってインストールしてください。

注記: 表示の問題について

NVIDIA ドライバをインストールすると、仮想マシンマネージャ はゲスト OS のディスプレイに接続できなくなります。ゲスト VM にアクセスするには、`ssh` 経由でログインするか、もしくはシリアルコンソールか専用の VNC サーバ経由でログインする必要があります。また画面の乱れを防ぐため、ディスプレイマネージャを停止して無効化しておく必要があります:

```
> sudo systemctl stop display-manager && systemctl disable display-manager
```

手順 A.3: LINUX ドライバの動作テスト

1. CUDA のサンプルテンプレートのあるディレクトリに移動します:

   ```
   > cd /usr/local/cuda-9.1/samples/0_Simple/simpleTemplates
   ```

2. あとは `simpleTemplates` ファイルをコンパイルして実行します:

   ```
   > make && ./simpleTemplates
   runTest<float,32>
   GPU Device 0: "Tesla V100-PCIE-16GB" with compute capability 7.0
   CUDA device [Tesla V100-PCIE-16GB] has 80 Multi-Processors
   Processing time: 495.006000 (ms)
   Compare OK
   runTest<int,64>
   GPU Device 0: "Tesla V100-PCIE-16GB" with compute capability 7.0
   CUDA device [Tesla V100-PCIE-16GB] has 80 Multi-Processors
   Processing time: 0.203000 (ms)
   Compare OK
   [simpleTemplates] -> Test Results: 0 Failures
   ```

A.4.2.2 Microsoft Windows ゲストの場合

重要

NVIDIA ドライバをインストールする前に、まずはゲスト側の `libvirt` 設定内に `<hidden state='on'/>` を追加して、ドライバからハイパーバイザを隠蔽する必要があります。たとえば下記のようになります:

```
<features>
 <acpi/>
 <apic/>
 <kvm>
  <hidden state='on'/>
```

```
  </kvm>
</features>
```

1. あとは https://www.nvidia.com/Download/index.aspx から NVIDIA ドライバをダウンロードして、インストールします。
2. また、https://developer.nvidia.com/cuda-downloads?target_os=Windows&target_arch=x86_64 から CUDA ツールキットをダウンロードしてインストールします。
3. インストールが完了すると、NVIDIA のデモサンプルがゲスト内の `Program Files\Nvidia GPU Computing Toolkit\CUDA\v10.2\extras\demo_suite` ディレクトリに展開されます。

B GNU ライセンス

本付録には、GNU Free Documentation License バージョン 1.2 とその日本語訳 (八田真行氏 (mhatta@gnu.org) による翻訳) を収録しています。

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU フリー文書利用許諾契約書

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. この利用許諾契約書を、一字一句そのままに複製し頒布することは許可する。しかし変更は認めない。

0. はじめに

この利用許諾契約書の目的は、この契約書が適用されるマニュアルや教科書、その他機能本位で実用的な文書を(無料ではなく)自由という意味で「フリー」とすること、すなわち、改変の有無あるいは目的の営利非営利を問わず、文書を複製し再頒布する自由をすべての人々に効果的に保証することです。加えてこの契約書により、著者や出版者が自分たちの著作物に対して相応の敬意と賞賛を得る手段も保護されます。また、他人が行った改変に対して責任を負わずに済むようになります。

この利用許諾契約書は「コピーレフト」的なライセンスの一つであり、この契約書が適用された文書から派生した著作物は、それ自身もまた原本と同じ意味でフリーでなければなりません。この契約書は、フリーソフトウェアのために設計されたコピーレフトなライセンスであるGNU一般公衆使用許諾契約書を補足するものです。

この利用許諾契約書は、フリーソフトウェア用のマニュアルに適用することを目的として書かれました。フリーソフトウェアはフリーな文書を必要としており、フリーなプログラムはそのソフトウェアが保証するのと同じ自由を提供するマニュアルと共に頒布されるべきだからです。しかし、この契約書の適用範囲はソフトウェアのマニュアルに留まりません。対象となる著作物において扱われる主題が何であれ、あるいはそれが印刷された書籍として出版されるか否かに関わらず、この契約書は文字で書かれたいかなる著作物にも適用することが可能です。私たちとしては、主にこの契約書を解説や参照を目的とする著作物に適用することをお勧めします。

1. この利用許諾契約書の適用範囲と用語の定義

著作物がこの利用許諾契約書の定める条件の下で頒布される旨の告知を、著作権者がその中に書いたすべてのマニュアルあるいはその他の著作物は、いかなる媒体上にあってもこの契約書の適用対象となる。そのような告知を置くことで、全世界において、著作権使用料を必要とせず、許可の存続期間を限定されること無く、この契約書の中で述べられている条件の下で当該著作物を利用できるという許可を与えることとする。以下において、「『文書』(Document)」とはそのような告知が記載されたマニュアルないし著作物すべてを指す。公衆の一員ならば誰でも契約の当事者となることができ、この契約書中では「あなた」と表現される。あなたは、著作権法の下で許可を必要とするような方法で著作物を複製や改変、あるいは頒布することにより、この契約書を受諾することになる。

『文書』の「改変版 (Modified Version)」とは、一字一句忠実に複製したか、あるいは改変や他言語への翻訳を行ったかどうかに関わらず、その『文書』の全体あるいは一部分を含む著作物すべてを意味する。

「補遺部分 (Secondary Section)」とは、『文書』中でその旨指定された補遺ないし本文に先だって前付けとして置かれる一部分であり、『文書』の出版者あるいは著者と、『文書』全体の主題(あるいはそれに関連する事柄)との関係のみを論じ、全体としての主題の範疇に直接属する内容を全く含まないものである(たとえば、『文書』の一部が数学の教科書だった場合、補遺部分では数学について何も解説してはならない)。補遺部分で扱われる関係は、その主題あるいは関連する事柄との歴史的なつながりのことかも知れないし、それらに関する法的、商業的、哲学的、倫理的、あるいは政治的立場についてかも知れない。

「変更不可部分 (Invariant Sections)」とは補遺部分の一種で、それらが変更不可部分であることが、『文書』をこの利用許諾契約書の下で発表する旨述べた告知中においてその部分の題名と共に明示されているものである。ある部分が上記のような「補遺」性の定義にそぐわない場合は、その部分を「変更不可」として指定することは認められない。『文書』は、変更不可部分を全く含まなくても良い。『文書』において変更不可部分が全く指定されていなければ、その『文書』に変更不可部分は存在しないということである。

「カバーテキスト(Cover Texts)」とは、『文書』がこの利用許諾契約書の指定する条件の下で発表される旨述べた告知において、「表カバーテキスト」あるいは「裏カバーテキスト」として列挙された短い文章のことを指す。表カバーテキストは最大で5語、裏カバーテキストは最大で25語までとする。

『文書』の「透過的」複製物とは、機械による読み取りが可能な『文書』の複製物のことを指す。透過的な複製物の文書形式は、その仕様が一般の人々に入手可能で、『文書』の内容を一般的なテキストエディタ、または(画素で構成される画像ならば)一般的なペイントプログラム、あるいは(図面ならば)いくつかの広く入手可能な製図エディタで簡単に改訂するのに適しており、なおかつテキストフォーマッタへの入力に適する(あるいはテキストフォーマッタへの入力に適する諸形式への自動的な変換に適する)ものでなければならない。透過的なファイル形式への複製であっても、マークアップ、あるいはマークアップの不在が読者によるそれ以降の改変をわざと邪魔し阻害するように仕組まれたものは透過的であるとは見做されない。ある画像形式が、相当量のテキスト文章を表現するために使われた場合、それは透過的ではない。透過的ではない複製は「非透過的」複製と呼ばれる。

透過的複製に適した形式の例としては、マークアップを含まないプレーンなASCII形式、Texinfo入力形式、LaTeX入力形式、一般に入手可能なDTDを用いたSGMLあるいはXML、または人間による改変を想定して設計された、標準に準拠したシンプルなHTMLやPostScript、PDFなどが挙げられる。透過的な画像形式の例には、PNGやXCF、JPGが含まれる。非透過な形式としては、独占的なワードプロセッサでのみ閲覧編集できる独占的なファイル形式、普通には入手できないDTDまたは処理系を使ったSGMLやXML、ある種のワードプロセッサが生成する、出力のみを目的とした機械生成のHTMLやPostScript、PDFなどが含まれる。

「題扉 (Title Page)」とは、印刷された書籍に於いては、実際の表紙自身のみならず、この利用許諾契約書が表紙に掲載することを義務づける文章や図などを、読みやすい形で載せるのに必要なだけの、表紙に引き続く数ページをも意味する。表紙に類するものが無い形式で発表される著作物においては、「題扉」とは本文の始まりに先だって、その著作物の題名が最も目立つ形で現れる場所の近くに置かれる文章のことを指す。

「XYZと題された (Entitled XYZ)」部分とは、『文書』において「XYZ」と名付けられた一部分であり、その題名は正確に「XYZ」であるか、「XYZ」を他の言語に翻訳した上でその後ろに「XYZ」をそのまま括弧で括ったものを含む記述のどちらかである(ここでの「XYZ」とは、この利用許諾契約書において以下で言 及される特定の部分名を意味している。例えば「謝辞 (Acknowledgements)」、「献辞 (Dedications)」、「推薦の辞 (Endorsements)」、「履歴 (History)」)。あなたが『文書』を改変する場合、そのような部分の「題名を保存する (Preserve the Title)」とは、「XYZと題された」部分として、ここでの定義に従い題名を残すということである。

『文書』は、「保証否認警告 (Warranty Disclaimers)」を、この利用許諾契約書が『文書』に適用されると述べた告知の次に含んでも良い。この種の保証否認警告は、この契約書からの言及という形で利用条件に含まれるものと解されるが、保証の否認に関することについてのみ有効とする。こういった保証否認警告で示しうるその他のいかなる含意も無効であり、この契約書の効能には何ら影響を持たない。

2. 逐語的に忠実な複製

この利用許諾契約書、著作権表示、この契約書が『文書』に適用される旨述べた告知の三つがすべての複製物に複製され、かつあなたがこの契約書で指定されている以外のいかなる条件も追加しない限り、あなたはこの『文書』を、商用であるか否かを問わずいかなる形でも複製頒布することができる。あなたは、あなたが作成あるいは頒布する複製物に対して、閲覧や再複製を技術的な手法によって妨害、規制してはならない。しかしながら、複製と引き換えに代価を得てもかまわない。あなたが相当量の複製物を頒布する際には、本契約書第3項で指定される条件にも従わなければならない。

またあなたは、上記と同じ条件の下で、複製物を貸与したり複製物を公に開示することができる。

3. 大量の複製

もしあなたが、『文書』の印刷された(あるいは通常は印刷された表紙を持つ媒体における)複製物を100部を超えて出版し、また『文書』の利用許諾告知がカバーテキストの掲載を要求している場合には、指定されたすべてのカバーテキストを、表カバーテキストは表表紙に、裏カバーテキストは裏表紙に、はっきりと読みやすい形で載せた表紙の中に複製物本体を綴じ込まなければならない。また、両方の表紙において、それらの複製物の出版者としてのあなたをはっきりとかつ読みやすい形で確認できなければならない。表表紙では『文書』の完全な題名を、題名を構成するすべての語が等しく目立つようにして、視認可能な形で示さなければならない。それらの情報に加えて、表紙に他の文章や図などを加えることは許可される。表紙のみを変更した複製物は、それが『文書』の題名を保存し上記の条件を満たす限り、ほかの点では逐語的に忠実な複製物 として扱われる。

もしどちらかの表紙に要求されるカバーテキストの量が多すぎて読みやすく収めることが不可能ならば、あなたはテキスト先頭の一文(あるいは適切に収まるだけ)を実際の表紙に載せ、続きは隣接したページに載せるべきである。

あなたが『文書』の「非透過的」複製物を100部を超えて出版あるいは頒布する場合、それぞれの非透過な複製物と一緒に機械で読み取り可能な透過的複製物を添付するか、それぞれの非透過な複製物(あるいはそれに付属する文書)中で、公にアクセス可能なコンピュータネットワーク上の所在地を記述しなければならない。その場所には、非透過な複製物と内容的に寸分違わず、余計なものが追加されていない完全な『文書』の透過的複製物が置かれ、またそこから、ネットワークを利用する一般公衆が、一般に標準的と考えられるネットワークプロトコルを使ってダウンロードすることができなければならない。もしあなたが後者の選択肢を選ぶならば、その版の非透過な複製物を公衆に(直接、あるいはあなたの代理人ないし小売業者が)最後に頒布してから最低1年間は、その透過的複製物が指定の場所でアクセス可能であり続けることを保証するよう、非透過な複製物の大量頒布を始める際に十分に慎重な手順を踏まなければならない。

これは要望であり必要条件ではないが、『文書』の著者に、『文書』の更新された版をあなたに提供する機会を与えるため、透過非透過を問わず大量の複製物を再頒布し始める前には彼らにきちんと連絡しておいてほしい。

4. 改変

『文書』の改変版を、この利用許諾契約書と細部まで同一の契約の下で発表する限り、すなわち原本の役割を改変版で置き換えた形での頒布と改変を、その複製物を所有するすべての人々に許可する限り、あなたは改変版を上記第2項および第3項が指定する条件の下で複製および頒布することができる。さらに、あなたは改変版において以下のことを行わなければならない。

A. 題扉に(もしあればその他の表紙にも)、『文書』お よび『文書』のそれ以前の版と見分けがつく題名を載せること(もし以前の 版があれば、『文書』の「履歴 (History)」の部分に列記されているはずである)。もし元の版の出版者から許可を得たならば、以前の版と同じ題名を 使っても良い。

B. 題扉に、改変版における改変を行った1人以上の人物か団体名を列記すること。あわせて元の『文書』の著者として、最低5人(もし5人以下ならばすべて)の主要著者を列記すること。ただし元の著者たちがこの条件を免除した場合は除く。

C. 題扉に、改変版の出版者名を出版者として記載すること。

D. 『文書』にあるすべての著作権表示を残すこと。

E. 他の著作権表示の近くに、あなたの改変に対する適当な著作権表示を追加すること。

F. 著作権表示のすぐ後に、改変版をこの契約書の条件の下で利用することを公衆に対して許可する告知を含めること。その形式はこの契約書の末尾にある付記で示されている。

G. 元の『文書』の利用許諾告知に書かれた、変更不可部分の完全な一覧と、要求されるカバーテキストとを、改変版の利用許諾告 知でもそのまま残すこと。

H. この契約書の、変更されていない複製物を含めること。

I. 「履歴 (History)」と題された部分とその題名を保存し、そこに改変版の、少なくとも題名、出版年、新しく変更した部分の著 者名、出版者名を、題扉に掲載するのと同じように記載した一項を加えること。もし『文書』中に「履歴」と題された部分が存在しない場合には、『文書』の題名、出版年、著者、出版者を題扉に掲載するのと同じように記載した部分を用意し、上記で述べたような、改変版を説明する一項を加えること。

J. 『文書』中に、『文書』の透過的複製物への公共的アクセスのために指定されたネットワーク的所在地が記載されていたならば、それを保存すること。同様に、その『文書』の元になった以前の版で指定されていたネットワーク的所在地も載っていたならば、それも保存すること。これらの情報は「履歴(History)」の部分に置いても良い。ただし、それが『文書』自身より少なくとも4年前に出版された著作物の情報であったり、あるいは改変版が参考にしている版の元々の出版者から許可を得たならば、その情報を削除してもかまわない。

K. 「謝辞 (Acknowledgement)」あるいは「献辞 (Dedication)」等と題されたいかなる部分も、その部分の題名を保存し、その部分の内容(各貢献者への謝意あるいは献呈の意)と語調を保存すること。

L. 『文書』の変更不可部分を、その本文および題名を変更せずに保存すること。章番号やそれに相当するものは部分の題名の一部とは見做さない。

M. 「推薦の辞 (Endorsement)」というような章名が題された部分はすべて削除すること。そのような部分を改変版に含めてはならない。

N. すでに存在する部分を「推薦の辞 (Endorsement)」と題されるように改名したり、題名の点で変更不可部分のどれかと衝突するように改名してはならない。

O. 保証否認警告を保存すること。

もし改変版に、補遺部分としての条件を満たし、かつ『文書』から複製物された文章や図などをいっさい含んでいない、前書き的な章あるいは付録が新しく含まれるならば、あなたは希望によりそれらの部分の一部あるいはすべてを変更不可と宣言することができる。変更不可を宣言するためには、それらの部分の題名を改変版の利用許諾告知中の変更不可部分一覧に追加すれば良い。これらの題名は他の章名とは全く別のものでなければならない。

含まれる内容が、さまざまな集団によるあなたの改変版に対する推薦の辞のみである限り、あなたは、「推薦の辞 (Endorsement)」と題された章を追加することができる。推薦の辞の例としては、ピアレビューの陳述、あるいは文書がある標準の権威ある定義としてその団体に承認されたという声明などがある。

あなたは、5語までの一文を表カバーテキストとして、25語までの文を裏表紙テキストとして、改変版のカバーテキスト一覧の末尾に加えることができる。一個人ないし一団体が直接(あるいは団体内で結ばれた協定によって)加えることができるのは、表カバーテキストおよび裏カバーテキストとしてそれぞれ一文ずつのみである。もし以前すでにその文書において、表裏いずれかの表紙にあなたの(またはあなたが代表する同じ団体内で為された協定に基づく)カバーテキストが含まれていたならば、あなたが新たに追加することはできない。し かしあなたは、その古い文を加えた以前の出版者から明示的な許可を得たならば、古い文を置き換えることができる。

『文書』の著者あるいは出版者は、この利用許諾契約書によって、彼らの名前を利用することを許可しているわけではない。彼らの名前を改変版の宣伝に使ったり、改変版への明示的あるいは黙示的な保証のために使うことを許可するものではない。

5. 文書の結合

あなたは、上記第4項において改変版に関して定義された条件の下で、この利用許諾契約書の下で発表された複数の文書を一つにまとめることができる。その際、原本となる文書にある変更不可部分を全て、改変せずに結合後の著作物中に含め、それらをあなたが統合した著作物の変更不可部分としてその利用許諾告知において列記し、かつ原本にある全ての保証否認警告を保存しなければならない。

結合後の著作物についてはこの契約書の複製物を一つ含んでいればよく、同一内容の変更不可部分が複数ある場合には一つで代用してよい。もし同じ題名だが内容の異なる変更不可部分が複数あるならば、そのような部分のそれぞれの題名の最後に、(もし分かっているならば)その部分の原著者あるいは出版者の名前で、あるいは他と重ならないような番号を括弧で括って記載することで、それぞれ見分けが付くようにしなければならない。結合後の著作物の利用許諾告知における変更不可部分の一覧においても、章の題名に同様の調整をすること。

結合後の著作物においては、あなたはそれぞれの原本の「履歴 (History)」と 題されたあらゆる部分をまとめて、「履歴 (History)」と題された一章にしなければならない。同様に、「謝辞 (Acknowledgements)」あるいは「献辞 (Dedications)」と題されたあらゆる部分もまとめなければならない。あなたは「推薦の辞 (Endowsements)」と題されたあらゆる部分も削除しなければならない。

6. 文書の収集

あなたは、この利用許諾契約書の下で発表された複数の文書で構成される収集著作物を作ることができる。その場合、それぞれの文書が逐語的に忠実に複製されることを保障するために他のすべての点でこの契約書の定める条件に従う限り、さまざまな文書中のこの契約書の個々の複製物を、収集著作物中に複製物を一つ含めることで代用することができる。

あなたは、このような収集著作物から文書を一つ取り出し、それをこの契約書の下で頒布することができる。ただしその際には、この契約書の複製物を抽出された文書に挿入し、またその他すべての点でこの文書の逐語的に忠実な複製に関してこの契約書が定める条件に従わなければならない。

7. 独立した著作物の集積

『文書』あるいはその派生物を、他の別の独立した文書あるいは著作物と一緒にし、一巻の記憶装置あるいは頒布媒体に収めた編集著作物は、編集に起因する著作権が編集著作物に含まれる個々の著作物がその利用者に許可した法的権利を制限するよう行使されない限り、「集積」著作物と呼ばれる。『文書』が集積著作物に含まれる場合、この契約書は、『文書』と共にまとめられた他の独立した著作物には、それら自身が『文書』の派生物で無い限り適用されることにはならない。

このような『文書』の複製物において、この利用許諾契約書の第3項によりカバーテキストの掲載が要求されている場合、『文書』の量が集積著作物全体の2分の1以下であれば、『文書』のカバーテキストは集積著作物中で『文書』そのものの周りを囲む中表紙、あるいは『文書』が電子的形式である場合には表紙の電子的等価物にのみ配置するだけでよい。その場合以外は、カバーテキストは集積著作物全体を取り巻く印刷された表紙に掲載されなければならない。

8. 翻訳

翻訳は改変の一種と見做すので、あなたは『文書』の翻訳をこの利用許諾契約書の第4項の定める条件の下で頒布することができる。変更不可部分を翻訳によって置き換えるには著作権者の特別許可を必要とするが、元の変更不可部分に追加する形で変更不可部分の全てないし一部の翻訳を含めることはかまわない。この契約書や『文書』中の利用許諾告知、保証否認警告すべての英語原本も含める限り、あなたはこの契約書、告知、警告の翻訳を含めることができる。契約書や告知、警告に関して翻訳と英語原本との間に食い違いが生じた場合、英語原本が優先される。

典型的な例として、『文書』のある部分が原文で「Acknowledgements」、「Dedications」、あるいは「History」と題されていた場合、実際の題名を変更するには、題名を保存する(この契約書の第1項)ための条件(同第4項)を満たすことが必要となる。

9. 契約の終了

この利用許諾契約書の下で明確に提示されている場合を除き、あなたは『文書』を複製、改変、サブライセンス、あるいは頒布してはならない。このライセンスで指定されている以外の、『文書』の複製、改変、サブライセンス、頒布に関するすべての企ては無効であり、この契約書によって保証されるあなたの権利を自動的に終結させることとなる。しかし、この契約書の下であなたから複製物ないし諸権利を得た個人や団体に関しては、そういった人々がこの契約書に完全に従ったままである限り、彼らに与えられた許諾は終結しない。

10. 将来における本利用許諾契約書の改訂

フリーソフトウェア財団は、時によってGNUフリー文書利用許諾契約書の新しい改訂版を出版することができる。そのような新版は現在の版と理念においては似たものになるであろうが、新たに生じた問題や懸念を解決するため細部においては違ったものになるだろう。詳しくは https://www.gnu.org/copyleft/ を参照せよ。

GNUフリー文書利用許諾契約書のそれぞれの版には、新旧の区別が付くようなバージョン番号が振られている。もし『文書』において、この契約書のある特定の版か「それ以降のどの版でも」適用して良いと指定されている場合、あなたはフリーソフトウェア財団から発行された(草稿として発表されたものを除く)指定の版かそれ以降の版のうちどれか一つを選び、その条項や条件に従うことができる。もし『文書』がこの契約書のバージョン番号を指定していない場合には、あなたはフリーソフトウェア財団から今までに出版された(草稿として発表されたものを除く)版のうちからどれか一つを選ぶことができる。

ADDENDUM: この利用許諾契約書をあなたの文書に適用するには

この利用許諾契約書をあなたが書いた文書に適用するには、この契約書の複製物一つを文書中に含め、以下に示す著作権表示と利用許諾告知を題扉のすぐ後に置いて下さい:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

(訳: Copyright (C) 西暦年 あなたの名前. この文書を、フリーソフトウェア財団発行のGNU フリー文書利用許諾契約書(バージョン1.2かそれ以降から一つを選択)が定める条件の下で複製、頒布、あるいは改変することを許可する。変更不可部分、表カバーテキスト、裏カバーテキストは存在しない。この利用許諾契約書の複製物は「GNU フリー文書利用許諾契約書」という章に含まれている。)

もし変更不可部分や表カバーテキスト、裏カバーテキストがあれば、「変更不可部分…は存在しない。」というところを以下で置き換えてください:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

(訳: (章の題名を列記)は変更不可部分であり、(表カバーテキストを列記)は表カバーテキスト、(裏カバーテキストを列記)は裏カバーテキストである。)

変更不可部分はあるがカバーテキストは存在しないなど、その他の三者の組み合せに関しては、状況に合わせて上記二つの選択肢を混ぜてください。

あなたの文書に、他に類を見ない独自のプログラムコードのサンプルが含まれる場合、フリーソフトウェアにおいてそのコードを利用することを許可するために、そういったサンプルに関してはこの利用許諾契約書と同時にGNU一般公衆許諾契約書のようなフリーソフトウェア向けライセンスのうちどれか一つを選択して適用してもよい、というような条件の下で発表することを推奨します。