



SUSE Linux Enterprise Server 15 SP2

虚拟化指南

虚拟化指南

SUSE Linux Enterprise Server 15 SP2

概述虚拟化技术，并介绍虚拟化的统一接口 libvirt，以及有关特定超级管理程序的详细信息。

出版日期：2024 年 12 月 12 日

<https://documentation.suse.com> 

版权所有 © 2006– 2024 SUSE LLC 和贡献者。保留所有权利。

根据 GNU 自由文档许可证 (GNU Free Documentation License) 版本 1.2 或（根据您的选择）版本 1.3 中的条款，在此授予您复制、分发和/或修改本文档的许可权限；本版权声明和许可证附带不可变部分。许可证版本 1.2 的副本包含在题为“GNU 自由文档许可证”的部分。

有关 SUSE 商标，请参见 <https://www.suse.com/company/legal/> 。所有其他第三方商标是其各自所有者的财产。商标符号（®、™ 等）代表 SUSE 及其关联公司的商标。星号 (*) 代表第三方商标。

本指南力求涵盖所有细节，但这不能确保本指南准确无误。SUSE LLC 及其关联公司、作者和译者对于可能出现的错误或由此造成的后果皆不承担责任。

目录

关于本手册 xviii

- 1 可用文档 xviii
- 2 提供反馈 xix
- 3 文档约定 xx
- 4 产品生命周期和支持 xxii
 - SUSE Linux Enterprise Server 支持声明 xxii • 技术预览 xxiii

I 简介 1

1 虚拟化技术 2

- 1.1 概述 2
- 1.2 虚拟化功能 3
- 1.3 虚拟化的优点 3
- 1.4 虚拟化模式 4
- 1.5 I/O 虚拟化 4

2 Xen 虚拟化简介 7

- 2.1 基本组件 7
- 2.2 Xen 虚拟化体系结构 8

3 KVM 虚拟化简介 10

- 3.1 基本组件 10
- 3.2 KVM 虚拟化体系结构 10

4	Linux 容器简介	12
5	虚拟化工具	13
5.1	虚拟化控制台工具	13
5.2	虚拟化 GUI 工具	14
6	安装虚拟化组件	18
6.1	安装 KVM	18
6.2	安装 Xen	18
6.3	安装容器	19
6.4	模式	19
6.5	安装 UEFI 支持	20
6.6	启用 KVM 中的嵌套式虚拟化支持	21
7	支持的主机、Guest 和功能	23
7.1	主机环境（超级管理程序）	23
7.2	Guest 环境	24
	半虚拟化驱动程序的提供	24
7.3	KVM 硬件要求	25
7.4	功能支持	26
	主机 (Dom0)	26
	半虚拟化 Guest	27
	全虚拟化 Guest	28
II	使用 libvirt 管理虚拟机	30
8	启动和停止 libvirtd	31
9	Guest 安装	33
9.1	基于 GUI 的 Guest 安装	33

- 9.2 在命令行中使用 **virt-install** 安装 35
- 9.3 高级 Guest 安装方案 38
 - 在 Windows Guest 上使用内存气球 38 • 在安装中包含附加产品 38

10 基本 VM Guest 管理 40

- 10.1 列出 VM Guest 40
 - 使用虚拟机管理器列出 VM Guest 40 • 使用 **virsh** 列出 VM Guest 41
- 10.2 通过控制台访问 VM Guest 41
 - 打开图形控制台 41 • 打开串行控制台 43
- 10.3 更改 VM Guest 的状态：启动、停止、暂停 44
 - 使用虚拟机管理器更改 VM Guest 的状态 44 • 使用 **virsh** 更改 VM Guest 的状态 45
- 10.4 保存和恢复 VM Guest 的状态 46
 - 使用虚拟机管理器保存/恢复 47 • 使用 **virsh** 保存和恢复 47
- 10.5 创建和管理快照 48
 - 术语 48 • 使用虚拟机管理器创建和管理快照 49 • 使用 **virsh** 创建和管理快照 51
- 10.6 删除 VM Guest 53
 - 使用虚拟机管理器删除 VM Guest 53 • 使用 **virsh** 删除 VM Guest 54
- 10.7 迁移 VM Guest 54
 - 迁移要求 54 • 使用虚拟机管理器进行迁移 56 • 使用 **virsh** 进行迁移 57 • 分步操作示例 58
- 10.8 监视 61
 - 使用虚拟机管理器进行监视 61 • 使用 **virt-top** 进行监视 62 • 使用 **kvm_stat** 进行监视 63

11 连接和授权 66

11.1 身份验证 66

libvirtd 身份验证 67 • VNC 身份验证 71

11.2 连接到 VM 主机服务器 75

非特权用户的“system”访问权限 77 • 使用虚拟机管理器管理连接 78

11.3 配置远程连接 79

基于 SSH 的远程隧道 (qemu+ssh 或 xen+ssh) 79 • 使用 x509 证书进行远程 TLS/SSL 连接 (qemu+tls 或 xen+tls) 79

12 管理储存 87

12.1 使用虚拟机管理器管理储存 89

添加储存池 90 • 管理储存池 93

12.2 使用 **virsh** 管理储存 95

列出池和卷 95 • 启动、停止和删除池 96 • 将卷添加到储存池 97 • 从储存池中删除卷 98 • 将卷挂接到 VM Guest 99 • 从 VM Guest 分离卷 100

12.3 使用 **virtlockd** 锁定磁盘文件和块设备 100

启用锁定 100 • 配置锁定 101

12.4 联机调整 Guest 块设备的大小 102

12.5 在主机与 Guest 之间共享目录（文件系统直通） 103

12.6 通过 **libvirt** 使用 **RADOS** 块设备 104

13 管理网络 106

13.1 网桥 106

使用 YaST 管理网桥 106 • 通过命令行管理网桥 107 • 使用 VLAN 接口 109

- 13.2 虚拟网络 110
 - 使用虚拟机管理器管理虚拟网络 111 • 使用 **virsh** 管理虚拟网络 115

14 使用虚拟机管理器配置虚拟机 122

- 14.1 计算机设置 123
 - 概览 123 • 性能 124 • 处理器 125 • 内存 126 • 引导选项 127
- 14.2 储存 128
- 14.3 控制器 129
- 14.4 网络 130
- 14.5 输入设备 132
- 14.6 视频 133
- 14.7 USB 重定向器 135
- 14.8 杂项 136
- 14.9 使用虚拟机管理器添加 CD/DVD-ROM 设备 137
- 14.10 使用虚拟机管理器添加软盘设备 138
- 14.11 使用虚拟机管理器弹出和更换软盘或 CD/DVD-ROM 媒体 139
- 14.12 将主机 PCI 设备指派到 VM Guest 140
 - 使用虚拟机管理器添加 PCI 设备 140
- 14.13 将主机 USB 设备指派到 VM Guest 141
 - 使用虚拟机管理器添加 USB 设备 141

15 使用 **virsh** 配置虚拟机 143

- 15.1 编辑 VM 配置 143
- 15.2 更改计算机类型 144

- 15.3 配置超级管理程序功能 145
- 15.4 配置 CPU 分配 146
- 15.5 更改引导选项 147
 - 更改引导顺序 147 • 使用直接内核引导 148
- 15.6 配置内存分配 149
- 15.7 添加 PCI 设备 149
- 15.8 添加 USB 设备 152
- 15.9 添加 SR-IOV 设备 153
 - 要求 153 • 装载和配置 SR-IOV 主机驱动程序 154 • 将 VF 网络设备添加到 VM Guest 158 • 动态分配池中的 VF 161
- 15.10 列出挂接的设备 162
- 15.11 配置储存设备 163
- 15.12 配置控制器设备 164
- 15.13 配置视频设备 166
 - 更改分配的 VRAM 量 166 • 更改 2D/3D 加速状态 167
- 15.14 配置网络设备 167
 - 使用多队列 virtio-net 提升网络性能 167
- 15.15 使用 Macvtap 共享 VM 主机服务器网络接口 168
- 15.16 禁用内存气球设备 169
- 15.17 配置多个监视器（双头） 169

16 使用 Vagrant 管理虚拟机 172

- 16.1 Vagrant 简介 172
 - Vagrant 概念 172 • Vagrant 示例 173
- 16.2 适用于 SUSE Linux Enterprise 的 Vagrant 盒子 173

16.3 更多资料 174

III 独立于超级管理程序的功能 175

17 磁盘缓存模式 176

17.1 磁盘接口缓存模式 176

17.2 缓存模式的说明 176

17.3 缓存模式对数据完整性的影响 178

17.4 缓存模式对性能的影响 178

17.5 缓存模式对实时迁移的影响 179

18 VM Guest 时钟设置 180

18.1 KVM: 使用 `kvm_clock` 180

其他计时方法 181

18.2 Xen 虚拟机时钟设置 181

19 libguestfs 182

19.1 VM Guest 操作概述 182

VM Guest 操作风险 182 • libguestfs 的设计用途 182

19.2 软件包安装 183

19.3 Guestfs 工具 184

修改虚拟机 184 • 支持的 filesystem 和磁盘映像 184 • **virt-**

rescue 184 • **virt-resize** 185 • 其他 `virt-*` 工

具 187 • **guestfish** 190 • 将物理机转换为 KVM Guest 191

19.4 查错 193

Btrfs 相关的问题 193 • 环境 193 • **libguestfs-test-tool** 194

19.5 外部参考信息 194

20 QEMU Guest 代理 195

- 20.1 运行 QEMU GA 命令 195
- 20.2 需要 QEMU GA 的 **virsh** 命令 195
- 20.3 增强 libvirt 命令 196
- 20.4 更多信息 197

IV 使用 XEN 管理虚拟机 198

21 设置虚拟机主机 199

- 21.1 最佳实践和建议 199
- 21.2 管理 Dom0 内存 200
 - 设置 Dom0 内存分配 200
- 21.3 全虚拟化 Guest 中的网卡 201
- 21.4 启动虚拟机主机 202
- 21.5 PCI 直通 203
 - 配置超级管理程序以使用 PCI 直通 204 · 将 PCI 设备指派给 VM Guest 系统 205 · VGA 直通 206 · 查错 206 · 更多信息 207
- 21.6 USB 直通 207
 - 标识 USB 设备 207 · 模拟的 USB 设备 208 · 半虚拟化 PVUSB 208

22 虚拟网络 211

- 22.1 Guest 系统的网络设备 211
- 22.2 Xen 中基于主机的路由 212
- 22.3 创建掩蔽网络设置 215
- 22.4 特殊注意事项 217
 - 虚拟网络中的带宽限制 217 · 监视网络流量 218

23 管理虚拟化环境 219

23.1 XL — Xen 管理工具 219

Guest 域配置文件 220

23.2 自动启动 Guest 域 221

23.3 事件操作 221

23.4 时戳计数器 222

23.5 保存虚拟机 223

23.6 恢复虚拟机 224

23.7 虚拟机状态 224

24 Xen 中的块设备 225

24.1 将物理储存映射到虚拟磁盘 225

24.2 将网络储存映射到虚拟磁盘 226

24.3 基于文件的虚拟磁盘和回写设备 226

24.4 调整块设备的大小 227

24.5 用于管理高级储存方案的脚本 228

25 虚拟化：配置选项和设置 229

25.1 虚拟 CD 读取器 229

半虚拟计算机上的虚拟 CD 读取器 229 • 全虚拟计算机上的虚拟 CD 读取器 229 • 添加虚拟 CD 读取器 230 • 去除虚拟 CD 读取器 231

25.2 远程访问方法 231

25.3 VNC 查看器 231

向虚拟机指派 VNC 查看器端口号 232 • 使用 SDL 而不是 VNC 查看器 233

25.4 虚拟键盘 233

- 25.5 分配专用 CPU 资源 234
 - Dom0 234 • VM Guest 235
- 25.6 HVM 功能 236
 - 指定引导时使用的引导设备 236 • 更改 Guest 的 CUID 236 • 增加 PCI-IRQ 的数量 237
- 25.7 虚拟 CPU 调度 238
- 26 管理任务 239**
 - 26.1 引导加载程序 239
 - 26.2 稀疏映像文件和磁盘空间 240
 - 26.3 迁移 Xen VM Guest 系统 241
 - 检测 CPU 功能 242 • 准备要迁移的块设备 243 • 迁移 VM Guest 系统 244
 - 26.4 监视 Xen 244
 - 使用 **xentop** 监视 Xen 244 • 其他工具 245
 - 26.5 提供 VM Guest 系统的主机信息 246
- 27 XenStore：在域之间共享的配置数据库 248**
 - 27.1 简介 248
 - 27.2 文件系统接口 248
 - XenStore 命令 249 • /vm 249 • /local/domain/<domid> 252
- 28 使用 Xen 作为高可用性虚拟化主机 254**
 - 28.1 使用远程储存实现 Xen HA 254
 - 28.2 使用本地储存实现 Xen HA 255
 - 28.3 Xen HA 和专用网桥 256

V 使用 QEMU 管理虚拟机 257

29 QEMU 概述 258

30 设置 KVM VM 主机服务器 259

30.1 CPU 的虚拟化支持 259

30.2 要求的软件 259

30.3 特定于 KVM 主机的功能 261

使用具有 virtio-scsi 的主机储存 262 • 使用 vhost-net 实现加速网络 263 • 使用多队列 virtio-net 提升网络性能 263 • VFIO：对设备进行安全的直接访问 265 • VirtFS：在主机与 Guest 之间共享目录 266 • KSM：在 Guest 之间共享内存页 268

31 Guest 安装 270

31.1 使用 **qemu-system-ARCH** 进行基本安装 270

31.2 使用 **qemu-img** 管理磁盘映像 271

有关 qemu-img 调用的一般信息 272 • 创建、转换和检查磁盘映像 273 • 使用 qemu-img 管理虚拟机的快照 279 • 有效操作磁盘映像 281

32 使用 **qemu-system-ARCH** 运行虚拟机 286

32.1 基本的 **qemu-system-ARCH** 调用 286

32.2 常规 **qemu-system-ARCH** 选项 286

基本虚拟硬件 287 • 储存和读取虚拟设备的配置 289 • Guest 实时时钟 290

32.3 在 QEMU 中使用设备 290

块设备 291 • 图形设备和显示选项 297 • USB 设备 299 • 字符设备 300

- 32.4 QEMU 中的网络 303
 - 定义网络接口卡 303 • 用户模式网络 304 • 桥接网络 306
- 32.5 使用 VNC 查看 VM Guest 309
 - 保护 VNC 连接 311

33 使用 QEMU 监视器管理虚拟机 314

- 33.1 访问监视器控制台 314
- 33.2 获取有关 Guest 系统的信息 315
- 33.3 更改 VNC 口令 318
- 33.4 管理设备 318
- 33.5 控制键盘和鼠标 319
- 33.6 更改可用内存 320
- 33.7 转储虚拟机内存 320
- 33.8 管理虚拟机快照 321
- 33.9 挂起和恢复虚拟机执行 322
- 33.10 实时迁移 323
- 33.11 QMP - QEMU 计算机协议 324
 - 通过标准输入/输出访问 QMP 324 • 通过 Telnet 访问 QMP 325 • 通过 Unix 套接字访问 QMP 326 • 通过 libvirt 的 **virsh** 命令访问 QMP 327

VI 使用 LXC 管理虚拟机 328

34 Linux 容器 329

- 34.1 设置 LXC 发行套件容器 329
- 34.2 设置 LXC 应用程序容器 332

- 34.3 使用 AppArmor 保护容器 333
- 34.4 libvirt LXC 驱动程序与 LXC 之间的差异 334
- 34.5 跨容器共享名称空间 335
- 34.6 更多信息 335

35 从 LXC 迁移到 libvirt-lxc 336

- 35.1 主机迁移 336
- 35.2 容器迁移 337
- 35.3 启动容器 338

词汇表 339

A 虚拟机驱动程序 349

B 附录 350

- B.1 安装半虚拟化驱动程序 350
 - 安装适用于 Microsoft Windows* 的 virtio 驱动程序 350

C XM、XL 工具堆栈和 Libvirt 框架 351

- C.1 Xen 工具堆栈 351
 - 从 xend/xm to xl/libxl 升级 351 · XL 设计 352 · 升级前的核对清单 352
- C.2 将 Xen 域配置导入 libvirt 353
- C.3 xm 与 xl 应用程序之间的差异 355
 - 表示法约定 355 · 新的全局选项 355 · 未更改的选项 356 · 已去除的选项 360 · 已更改的选项 363 · 新选项 376
- C.4 外部链接 377

C.5 以 **xm** 兼容的格式保存 Xen Guest 配置 378

D GNU 许可证 379

关于本手册

本手册介绍如何在 SUSE Linux Enterprise Server 上使用 KVM（基于内核的虚拟机）、Xen 和 Linux 容器 (LXC) 设置和管理虚拟化。第一部分介绍几种不同的虚拟化解决方案，其中描述了它们的要求、安装和 SUSE 的支持状态。第二部分介绍如何使用 `libvirt` 管理 VM Guest 和 VM 主机服务器。接下来的部分介绍各种管理任务和实践，最后三个部分介绍特定于超级管理程序的主题。

1 可用文档



注意：联机文档和最新更新

我们的产品文档可从 <https://documentation.suse.com/> 获取，您也可以在此处找到最新更新，以及浏览或下载各种格式的文档。最新的文档更新通常会在文档的英文版中提供。

针对本产品提供的文档如下：

《安装快速入门》文章

本《快速入门》引导您逐步完成安装 SUSE® Linux Enterprise Server 15 SP2 的过程。

《部署指南》

此指南详细介绍如何安装单个或多个系统，以及如何利用产品继承功能部署基础结构。

从各种方法中选择：从物理安装媒体进行本地安装；自定义标准安装映像；网络安装服务器；使用远程控制的高度自定义的自动化安装过程进行大规模部署；及初始系统配置。

《管理指南》

讲述系统管理任务，如维护、监视和自定义初始安装的系统。

《虚拟化指南》

概述虚拟化技术，并介绍虚拟化的统一接口 `libvirt`，以及有关特定超级管理程序的详细信息。

《储存管理指南》

提供有关如何在 SUSE Linux Enterprise Server 上管理储存设备的信息。

《AutoYaST 指南》

AutoYaST 系统使用包含安装和配置数据的 AutoYaST 配置文件，让您以无人照管方式批量部署 SUSE Linux Enterprise Server 系统。该手册将引导您完成自动安装的基本步骤，包括准备、安装和配置。

《安全指南》

介绍系统安全的基本概念，包括本地安全方面和网络安全方面。说明如何使用产品固有的安全软件（例如 AppArmor），或者能够可靠收集有关任何安全相关事件的信息的审核系统。

《强化指南》

处理安装和设置安全 SUSE Linux Enterprise Server 的特定事项以及进一步确保和强化安装所需的额外安装后步骤。支持管理员选择与安全相关的选项并做出决策。

《系统分析和微调指南》

关于问题检测、解决和优化的管理员指南。了解如何使用监视工具检查和优化系统以及如何有效管理资源。还包含常见问题和解决方法的概述以及其他帮助和文档资源。

《储存库镜像工具指南》

订阅管理工具管理员指南。订阅管理工具是用于 SUSE Customer Center 并包含储存库和注册目标的代理系统。了解如何安装和配置本地 SMT 服务器、镜像和管理储存库、管理客户端计算机，以及配置客户端以使用 SMT。

《GNOME 用户指南》

介绍 SUSE Linux Enterprise Server 的 GNOME 桌面。指导您使用和配置桌面并帮助您执行关键任务。它主要面向想要有效使用 GNOME 作为其默认桌面的最终用户。

<https://www.suse.com/releases/notes/> 上提供了本产品的发行说明。

2 提供反馈

欢迎您提供针对本文档的反馈及改进建议！我们提供了多种反馈渠道：

服务请求和支持

有关产品可用的服务和支持选项，请参见 <https://www.suse.com/support/>。

要创建服务请求，需在 SUSE Customer Center 中获取一个订阅。请转到 <https://scc.suse.com/support/requests> 并登录，然后单击新建。

Bug 报告

在 <https://bugzilla.suse.com/> 中报告文档问题。要简化此过程，可以使用本文档 HTML 版本中的标题旁边的报告文档 Bug 链接。这样，就会在 Bugzilla 中预先选择正确的产品和类别，并添加当前章节的链接。然后，您便可以立即开始键入 Bug 报告。需要一个 Bugzilla 帐户。

贡献

要帮助改进本文档，请使用本文档 HTML 版本中的标题旁边的编辑源代码链接。这些链接会将您转到 GitHub 上的源代码，在其中可以创建拉取请求。需要一个 GitHub 帐户。

有关本文档使用的文档环境的详细信息，请参见[储存库的 README \(https://github.com/SUSE/doc-sle/blob/master/README.adoc\)](https://github.com/SUSE/doc-sle/blob/master/README.adoc)。

邮件

另外，您也可以将有关本文档中的错误以及相关反馈发送至：doc-team@suse.com。

请确保反馈中含有文档标题、产品版本和文档发布日期。请引用相关的章节号和标题（或者包含 URL），并提供问题的简要说明。

3 文档约定

本文档中使用了以下通知和排版约定：

- /etc/passwd：目录名称和文件名
- PLACEHOLDER：PLACEHOLDER 将会替换为实际的值
- PATH：环境变量 PATH
- ls、--help：命令、选项和参数
- user：用户和组
- package name：软件包名称
- **Alt**、**Alt - F1**：按键或组合键；这些键以大写形式显示，如在键盘上一样
- 文件、文件 > 另存为：菜单项，按钮

- **AMD/Intel** 本段内容仅与 AMD64/Intel 64 体系结构相关。箭头标记文本块的开始位置和结束位置。 ◁
- **IBM Z, POWER** 本段内容仅与 IBM Z 和 POWER 体系结构相关。箭头标记文本块的开始位置和结束位置。 ◁
- 跳舞的企鹅（企鹅一章，↑其他手册）：此内容参见自其他手册中的一章。
- 必须使用 root 特权运行的命令。您往往还可以在这些命令前加上 sudo 命令，以非特权用户身份来运行它们。

```
root # command
tux > sudo command
```

- 可以由非特权用户运行的命令。

```
tux > command
```

- 注意



警告：警告通知

在继续操作之前，您必须了解的不可或缺的信息。向您指出有关安全问题、潜在数据丢失、硬件损害或物理危害的警告。



重要：重要通知

在继续操作之前，您必须了解的重要信息。



注意：注意通知

额外信息，例如有关软件版本差异的信息。



提示：提示通知

有用信息，例如指导方针或实用性建议。

4 产品生命周期和支持

SUSE 产品的支持周期长达 13 年。要查看产品的生命周期日期，请参见 <https://www.suse.com/lifecycle/>。

SUSE Linux Enterprise 适用以下生命周期和发行周期：

- SUSE Linux Enterprise Server 的生命周期为 13 年：10 年的标准支持，3 年的扩展支持。
- SUSE Linux Enterprise Desktop 的生命周期为 10 年：7 年的标准支持，3 年的扩展支持。
- 主要版本每 4 年发行一次。服务包每 12-14 个月发行一次。
- 新的 SUSE Linux Enterprise 服务包发行后，SUSE 对以前的服务包的支持会延续 6 个月。

某些产品提供长期服务包支持 (LTSS)。有关我们的支持政策和选项的信息，请参见 <https://www.suse.com/support/policy.html> 和 <https://www.suse.com/support/programs/long-term-service-pack-support.html>。

模块的生命周期、更新策略和更新时限不同于其基础产品。模块包含软件包，是完全受到支持的 SUSE Linux Enterprise Server 组件。有关详细信息，请参见《模块和扩展快速入门》文章。

4.1 SUSE Linux Enterprise Server 支持声明

要获得支持，您需要一个适当的 SUSE 订阅。要查看为您提供的具体支持服务，请转到 <https://www.suse.com/support/> 并选择您的产品。

支持级别的定义如下：

L1

问题判定，该技术支持级别旨在提供兼容性信息、使用支持、持续维护、信息收集，以及使用可用文档进行基本查错。

L2

问题隔离，该技术支持级别旨在分析数据、重现客户问题、隔离问题领域，并针对级别 1 不能解决的问题提供解决方法，或作为级别 3 的准备级别。

L3

问题解决，该技术支持级别旨在借助工程方法解决级别 2 支持所确定的产品缺陷。

对于签约的客户与合作伙伴，SUSE Linux Enterprise Server 将为除以下包外的其他所有包提供 L3 支持：

- 技术预览。
- 声音、图形、字体和作品。
- 需要额外客户合同的软件包。
- 工作站扩展模块随附的某些软件包仅享受 L2 支持。
- 名称以 `-devel` 结尾的包（包含头文件和类似的开发人员资源）只能同其主包一起接受支持。

SUSE 仅支持使用原始包，即，未发生更改且未重新编译的包。

4.2 技术预览

技术预览是 SUSE 提供的旨在让用户大致体验未来创新的各种包、堆栈或功能。随附这些预览只是为了提供方便，让您有机会在自己的环境中测试新的技术。非常希望您能提供反馈！如果您测试了技术预览，请联系 SUSE 代表，将您的体验和用例告知他们。您的反馈对于我们的未来开发非常有帮助。

但是，技术预览存在以下限制：

- 技术预览仍处于开发阶段。因此，它们的功能可能不完备、不稳定，或者在其他方面不适合用于生产。
- 技术预览不受支持。
- 技术预览可能仅适用于特定的硬件体系结构。
- 技术预览的细节和功能可能随时会发生变化。因此，可能无法升级到技术预览的后续版本，而只能进行全新安装。
- 我们随时可能会放弃技术预览。例如，如果 SUSE 发现某个预览不符合客户或市场的需求，或者不能证明它符合企业标准，则可能会放弃该预览。SUSE 不承诺未来将提供此类技术的受支持版本。

有关产品随附的技术预览的概述，请参见 <https://www.suse.com/releasesnotes/>  上的发行说明。

| 简介

- 1 虚拟化技术 2
- 2 Xen 虚拟化简介 7
- 3 KVM 虚拟化简介 10
- 4 Linux 容器简介 12
- 5 虚拟化工具 13
- 6 安装虚拟化组件 18
- 7 支持的主机、Guest 和功能 23

1 虚拟化技术

虚拟化技术提供了一种供计算机（主机）在主机操作系统上运行另一个操作系统（Guest 虚拟机）的方式。

1.1 概述

SUSE Linux Enterprise Server 中包含最新的开源虚拟化技术：Xen 和 KVM。借助这些超级管理程序，可以使用 SUSE Linux Enterprise Server 在单个物理系统上供应、取消供应、安装、监视和管理多个虚拟机 (VM Guest)（有关详细信息，请参见[超级管理程序](#)）。

SUSE Linux Enterprise Server 原生就能创建可运行经过修改、高度优化的半虚拟化操作系统，以及未经修改的全虚拟化操作系统的虚拟机。全虚拟化使 Guest 操作系统无需经过修改即可运行，但要求配备支持 Intel* 虚拟化技术 (Intel VT) 或 AMD* 虚拟化 (AMD-V) 的 AMD64/Intel 64 处理器。

操作系统中实现虚拟化的主要组件是超级管理程序（或虚拟机管理器），它是直接在服务器硬件上运行的软件层。超级管理程序控制着平台资源，并通过向每个 VM Guest 提供虚拟化的硬件接口，在多个 VM Guest 及其操作系统之间共享这些资源。

SUSE Linux Enterprise 是企业级 Linux 服务器操作系统，提供两种类型的超级管理程序：Xen 和 KVM。

这两种超级管理程序均支持 AMD64/Intel 64 体系结构上的虚拟化。对于 POWER 体系结构，支持 KVM。Xen 和 KVM 均支持全虚拟化模式。此外，Xen 支持半虚拟化模式，您可以在同一台主机上同时运行半虚拟化和全虚拟化 Guest。

包含 Xen 或 KVM 的 SUSE Linux Enterprise Server 充当虚拟化主机服务器 (VHS)，支持具有各自 Guest 操作系统的 VM Guest。SUSE VM Guest 体系结构由一个超级管理程序以及多个管理组件组成，这些管理组件构成运行许多托管了应用程序的 VM Guest 的 VHS。

在 Xen 中，管理组件在通常称为 **Dom0** 的特权 VM Guest 中运行。在 Linux 内核充当超级管理程序的 KVM 中，管理组件直接在 VHS 上运行。

1.2 虚拟化功能

虚拟化设计可为您的组织提供许多功能。操作系统虚拟化已广泛应用到众多计算领域中：

- 服务器整合：可用一台大型物理服务器取代许多服务器，以便整合硬件，并将 Guest 操作系统转换为虚拟机。服务器整合提供在新式硬件上运行传统软件的能力。
- 隔离：Guest 操作系统可与运行它的主机完全隔离。因此，如果虚拟机损坏，主机系统将不会受损。
- 迁移：将运行中的虚拟机转移到另一台物理机的过程。实时迁移是一项扩展功能，可以在不与客户端或应用程序断开连接的情况下实现这种转移。
- 灾难恢复：虚拟化 Guest 对硬件的依赖性不大，并且主机服务器会提供快照功能，以便在不造成任何损坏的情况下恢复已知可正常运行的系统。
- 动态负载平衡：让您方便地对整个基础结构中的服务进行负载平衡的一项迁移功能。

1.3 虚拟化的优点

虚拟化不仅能够提供与硬件服务器相同的服务，而且还具有许多优势。

首先，它降低了基础结构的成本。服务器主要用于向客户提供服务，而虚拟化的操作系统能够提供相同的服务，同时还具备以下优势：

- 更少的硬件：您可以在一台主机上运行多个操作系统，因此可以减轻总体硬件维护工作量。
- 更低的能耗/散热成本：更少的硬件意味着当您需要更多服务时，无需在电能、后备电源和散热方面投入更多成本。
- 节省空间：由于不需要更多的硬件服务器（服务器数量比运行的服务数量更少），因此可以节省数据中心的空间。
- 更少的管理工作：使用 VM Guest 可以简化基础结构的管理。
- 灵活性和工作效率：虚拟化提供迁移功能：实时迁移和快照。这些功能减少了停机时间，并且可以在不造成任何服务中断的情况下，让您轻松将服务从一个位置转移到另一个位置。

1.4 虚拟化模式

Guest 操作系统以全虚拟化 (FV) 模式或半虚拟 (PV) 模式托管在虚拟机上。每种虚拟化模式都有其优缺点。

- 全虚拟化模式允许虚拟机运行未经修改的操作系统，例如 Windows* Server 2003。它可以使用二进制转换或[硬件辅助](#)虚拟化技术，例如 AMD* 虚拟化或 Intel* 虚拟化技术。在支持此功能的处理器上使用硬件辅助可以提高性能。

以全虚拟化模式托管的某些 Guest 操作系统可配置为使用 SUSE 虚拟机驱动程序包 (VMDB) 中的驱动程序，而不是源自操作系统的驱动程序。在 Windows Server 2003 这样的 Guest 操作系统上运行虚拟机驱动程序可以大幅提升性能。有关更多信息，请参见[附录 A “虚拟机驱动程序”](#)。

- 要使 Guest 操作系统能够在半虚拟模式下运行，通常需要根据虚拟化环境对其进行修改。不过，以半虚拟模式运行的操作系统比全虚拟化模式下运行的操作系统的性能更佳。当前已经过修改可在半虚拟模式下运行的操作系统称为半虚拟化操作系统，包括 SUSE Linux Enterprise Server 和 NetWare® 6.5 SP8。

1.5 I/O 虚拟化

VM Guest 不仅可以共享主机系统的 CPU 和内存资源，还能共享 I/O 子系统的此类资源。由于软件 I/O 虚拟化技术提供的性能低于裸机，因此最近开发了接近“本机”性能的硬件解决方案。SUSE Linux Enterprise Server 支持以下 I/O 虚拟化技术：

全虚拟化

全虚拟化 (FV) 驱动程序会模拟受到广泛支持的真实设备，可以通过 VM Guest 中的现有驱动程序来使用这些设备。Guest 也称为硬件虚拟机 (HVM)。由于 VM 主机服务器上的物理设备可能不同于模拟的设备，超级管理程序需要先处理所有 I/O 操作，然后才能将其转交到物理设备。因此，所有 I/O 操作需要遍历两个软件层，这一过程不仅会显著影响 I/O 性能，而且还会消耗 CPU 时间。

半虚拟化

半虚拟化 (PV) 支持在超级管理程序与 VM Guest 之间直接通讯。与全虚拟化相比，它产生的开销更少，但性能却好很多。但使用半虚拟化技术时，无论是要支持半虚拟化 API 还是半虚拟化驱动程序，都必须修改 Guest 操作系统。有关支持半虚拟化的 Guest 操作系统列表，请参见第 7.2.1 节 “半虚拟化驱动程序的提供”。

PVHVM

这种类型的虚拟化通过半虚拟化 (PV) 驱动程序以及 PV 中断和计时器处理增强了 HVM（请参见全虚拟化）。

VFIO

VFIO 是虚拟功能 I/O 的英文缩写，是适用于 Linux 的新式用户级驱动程序框架。它取代了传统的 KVM PCI 直通设备指派。VFIO 驱动程序会在受安全内存 (IOMMU) 保护的环境中向用户空间公开直接的设备访问。利用 VFIO，VM Guest 可以直接访问 VM 主机服务器上的硬件设备（直通），避免性能关键型路径中的模拟操作造成性能问题。此方法不允许共享设备 — 每个设备只能指派到一个 VM Guest。VFIO 需受 VM 主机服务器 CPU、芯片组和 BIOS/EFI 的支持。

与传统的 KVM PCI 设备指派相比，VFIO 具有以下优势：

- 资源访问与安全引导兼容。
- 设备会被隔离，并且其内存访问受到保护。
- 提供设备所有权模型更为灵活的用户空间设备驱动程序。
- 独立于 KVM 技术，不局限于 x86 体系结构。

从 SUSE Linux Enterprise Server 12 SP2 开始，USB 和 PCI 直通设备指派方法被认为已过时，已由 VFIO 模型取代。

SR-IOV

作为最新的 I/O 虚拟化技术，单根 I/O 虚拟化 (SR-IOV) 结合了上述技术的优点 — 在兼顾性能的同时还能与多个 VM Guest 共享设备。SR-IOV 要求使用特殊的 I/O 设备，这些设备必须能够复制资源，使它们看似是多个独立设备。每个这样的“伪”设备可由单个 Guest 直接使用。但对于网卡（举例而言），可使用的并发队列数有限，因此与半虚拟化驱动程序相比，SR-IOV 有可能会降低 VM Guest 的性能。在 VM 主机服务器上，SR-IOV 必须受 I/O 设备、CPU 和芯片组、BIOS/EFI 及超级管理程序的支持 — 有关设置说明，请参见第 14.12 节 “将主机 PCI 设备指派到 VM Guest”。

! 重要：VFIO 和 SR-IOV 的要求

要能够使用 VFIO 和 SR-IOV 功能，VM 主机服务器需要满足以下要求：

- 需在 BIOS/EFI 中启用 IOMMU。
- 对于 Intel CPU，需在内核命令行上提供内核参数 `intel_iommu=on`。有关更多信息，请参见《管理指南》，第 14 章 “引导加载程序 GRUB 2”，第 14.3.3.2 节 “内核参数选项卡”。
- VFIO 基础结构需可用。装载内核模块 `vfio_pci` 即可做到这一点。有关更多信息，请参见《管理指南》，第 15 章 “systemd 守护程序”，第 15.6.4 节 “装载内核模块”。

2 Xen 虚拟化简介

本章介绍并阐述您在设置和管理基于 Xen 的虚拟化环境时需要了解的组件与技术。

2.1 基本组件

基于 Xen 的虚拟化环境的基本组件包括 Xen 超级管理程序、Dom0、任意数量的其他 VM Guest、工具、命令，以及用于管理虚拟化的配置文件。运行所有这些组件的物理计算机系统称为 VM 主机服务器，因为这些组件共同构成了托管虚拟机的平台。

Xen 超级管理程序

Xen 超级管理程序（有时简称为虚拟机监视器）是一个开源软件程序，用于协调虚拟机与物理硬件之间的低级别交互。

Dom0

虚拟机主机环境（也称为 Dom0 或控制域）由多个组件构成，其中包括：

- SUSE Linux Enterprise Server，提供图形环境和命令行环境，用于管理虚拟机主机组件及其虚拟机。



注意

术语“Dom0”指的是提供管理环境的特殊域。Dom0 能以图形模式或命令行模式运行。

- 基于 xenlight 库 (libxl) 的 xl 工具堆栈，用于管理 Xen Guest 域。
- 开源软件 QEMU，可模拟完整的计算机系统，包括处理器和各种外设。它提供以全虚拟化或半虚拟化模式托管操作系统的功能。

基于 Xen 的虚拟机

基于 Xen 的虚拟机（也称为 VM Guest 或 DomU）由以下组件构成：

- 至少一个包含可引导操作系统的虚拟磁盘。虚拟磁盘可以基于文件、分区、卷或其他类型的块设备。
- 每个 Guest 域的配置文件。这是遵循手册页 [man 5 xl.conf](#) 中所述语法的文本文件。
- 与控制域所提供的虚拟网络连接的多个网络设备。

管理工具、命令和配置文件

您可以结合使用 GUI 工具、命令与配置文件来管理和自定义虚拟化环境。

2.2 Xen 虚拟化体系结构

下图描绘了包含四个虚拟机的虚拟机主机。所示的 Xen 超级管理程序直接在物理硬件平台上运行。请注意，控制域也是虚拟机，不过相比所有其他虚拟机，它还承担了多项其他管理任务。

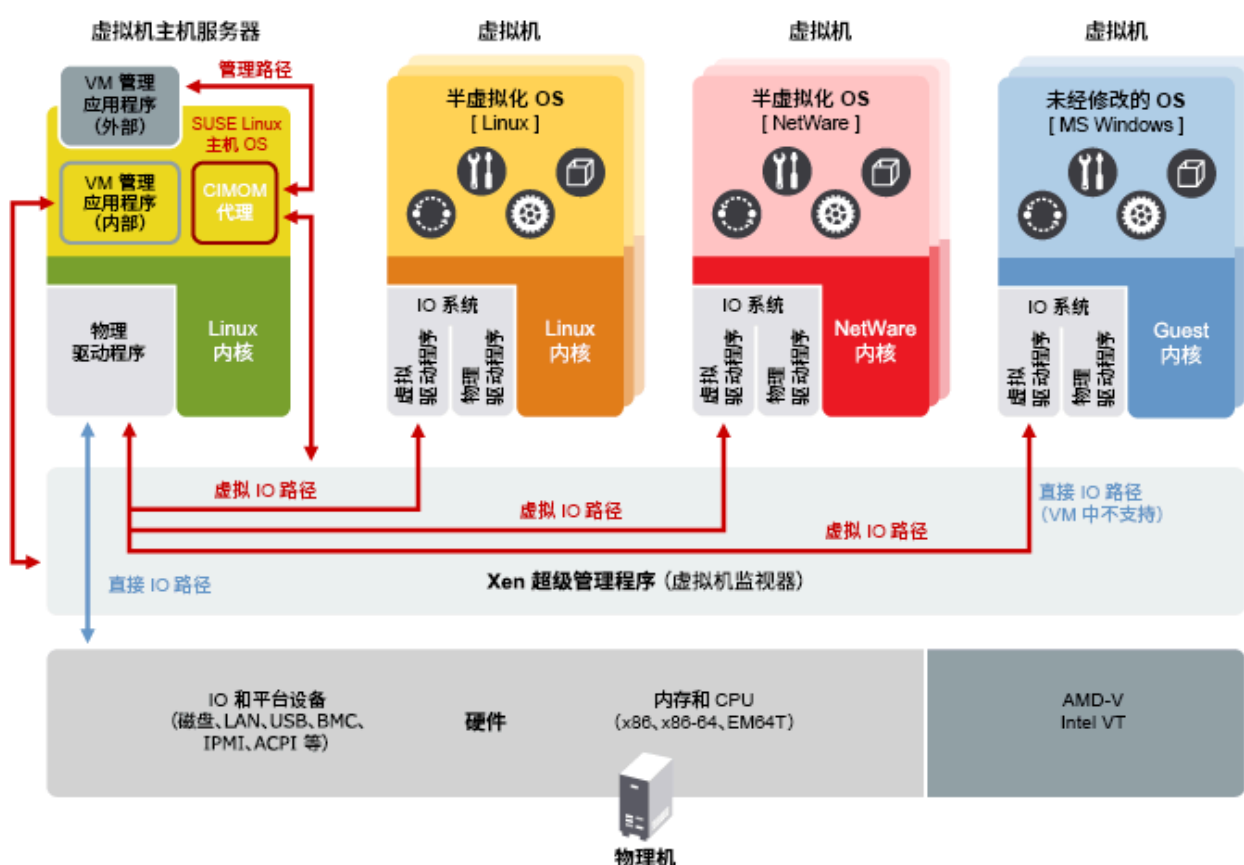


图 2.1：XEN 虚拟化体系结构

左侧所示虚拟机主机的 Dom0 运行的是 SUSE Linux Enterprise Server 操作系统。中间所示的两个虚拟机运行的是半虚拟化操作系统。右侧所示的虚拟机是全虚拟计算机，运行的是未经修改的操作系统，例如最新版本的 Microsoft Windows/Server。

3 KVM 虚拟化简介

3.1 基本组件

KVM 是适用于支持硬件虚拟化的 AMD64/Intel 64 和 IBM Z 体系结构的全虚拟化解决方案。

可以直接使用 QEMU 工具或使用基于 `libvirt` 的堆栈来管理 VM Guest（虚拟机）、虚拟储存和虚拟网络。QEMU 工具包括 `qemu-system-ARCH`、QEMU 监视器、`qemu-img` 和 `qemu-ndb`。基于 `libvirt` 的堆栈包括 `libvirt` 本身，以及 `virsh`、`virt-manager`、`virt-install` 和 `virt-viewer` 等基于 `libvirt` 的应用程序。

3.2 KVM 虚拟化体系结构

这款全虚拟化解决方案包括两个主要组件：

- 一组内核模块（`kvm.ko`、`kvm-intel.ko` 和 `kvm-amd.ko`），提供核心虚拟化基础结构和特定于处理器的驱动程序。
- 一个用户空间程序（`qemu-system-ARCH`），提供虚拟设备模拟以及用于管理 VM Guest（虚拟机）的控制机制。

术语 KVM 更适合表示内核级虚拟化功能，但在实践中，更多的是使用它来表示用户空间组件。

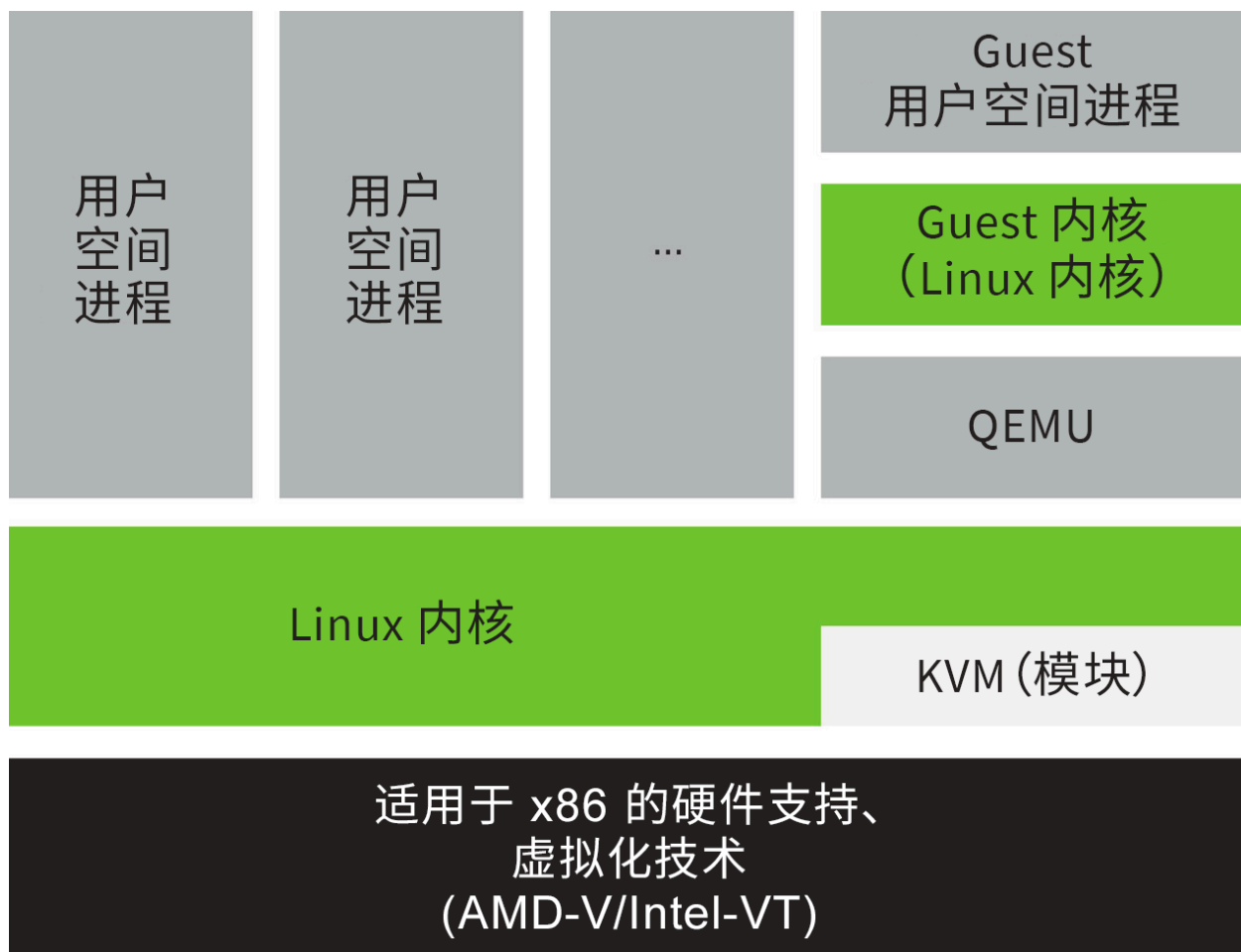


图 3.1：KVM 虚拟化体系结构



注意：Hyper-V 模拟支持

QEMU 可为 Windows* Guest 提供特定的 Hyper-V 超级调用来部分模拟 Hyper-V 环境。利用此项支持可以改进启用了 Hyper-V 的 Windows* Guest 的行为。

4 Linux 容器简介

Linux 容器提供了轻量级虚拟化方法，可在单台主机上同时运行多个虚拟环境（容器）。这与 `chroot` 环境类似。容器通过内核控制组 (`cgroup`) 和内核名称空间进行隔离。

容器提供操作系统级别的虚拟化，在此级别由内核控制隔离的容器。此方法与 Xen 或 KVM 等全虚拟化解决方案不同，在后者中，处理器将模拟整个硬件环境并控制虚拟机。

从概念上讲，可将容器视为一种改进的 `chroot` 技术。`chroot` 环境仅隔离文件系统，而容器可以通过 `cgroup` 进一步提供资源管理和控制。

使用容器的优势

- 通过自包含的单元隔离应用程序。
- 由于容器会实时管理资源分配，因此可提供接近本机的性能。
- 通过 `cgroup` 控制网络接口并在容器内部应用资源。

容器的限制

- 容器在主机系统内核的内部运行，因此无法使用不同的内核或不同的内核版本。
- 只有基于 Linux 的应用程序才可容器化。
- 容器不安全，总体安全性取决于主机系统。可以通过 AppArmor 或 SELinux 配置文件保护容器化应用程序。

5 虚拟化工具

libvirt 是一个库，提供用于管理 KVM、LXC、Xen 等流行虚拟化解决方案的通用 API。该库为这些虚拟化解决方案提供规范化管理 API，以便为更高层级的管理工具提供一个跨超级管理程序的稳定接口。该库还提供用于管理 VM 主机服务器上的虚拟网络和储存的 API。每个 VM Guest 的配置都储存在 XML 文件中。

您还可以使用 libvirt 来远程管理 VM Guest。它支持 TLS 加密、x509 证书和 SASL 身份验证。这样，您便可以通过单个工作站集中管理 VM 主机服务器，无需再单独访问每台 VM 主机服务器。

建议您使用基于 libvirt 的工具来管理 VM Guest。libvirt 与基于 libvirt 的应用程序之间的互操作性已经过测试，SUSE 的支持原则将其视为不可或缺的一部分。

5.1 虚拟化控制台工具

SUSE Linux Enterprise Server 提供了以下适用于命令行的基于 libvirt 的工具：

virsh (软件包： libvirt-client)

用于管理 VM Guest 的命令行工具，其功能与虚拟机管理器类似。可让您更改 VM Guest 的状态（启动、停止、暂停等）、设置新的 Guest 和设备，或编辑现有配置。virsh 还可用于编写 VM Guest 管理操作的脚本。

virsh 将第一个参数作为命令，将后续参数作为此命令的选项：

```
virsh [-c URI] COMMAND DOMAIN-ID [OPTIONS]
```

与 zypper 一样，您也可以调用不带命令的 virsh。在此情况下，virsh 会启动一个外壳并等待您发出命令。此模式非常适合必须运行后续命令的情形：

```
~> 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 (软件包: virt-install)

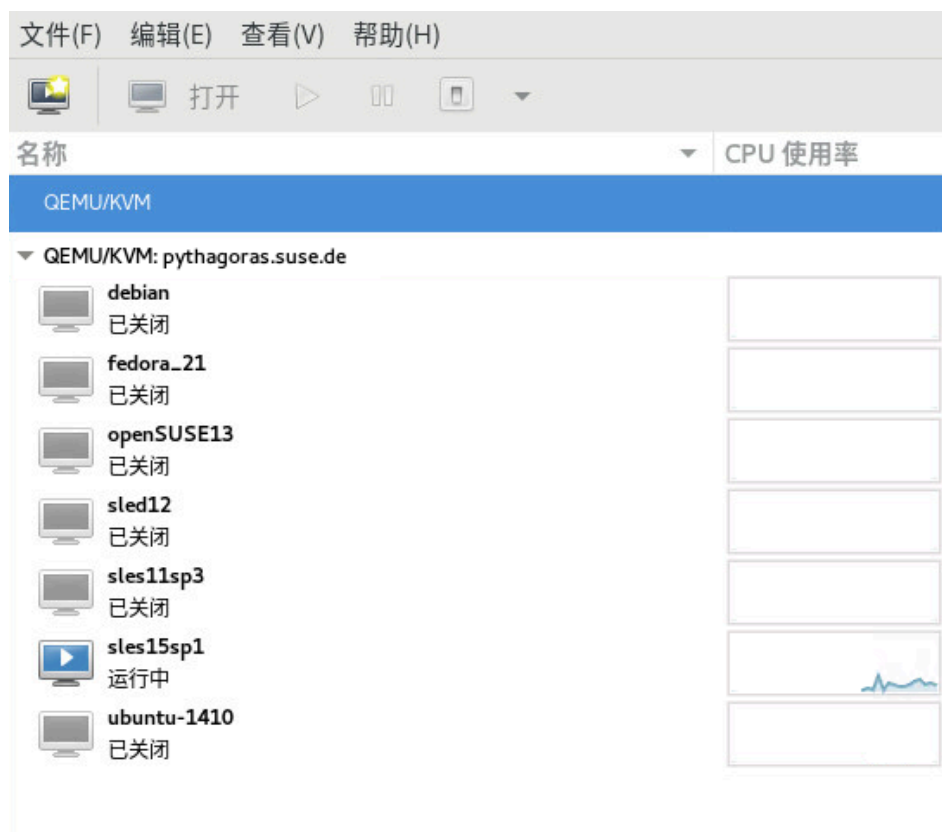
用于通过 libvirt 库创建新 VM Guest 的命令行工具。它支持通过 VNC 或 **SPICE** 协议进行图形安装。如果指定了适当的命令行参数，**virt-install** 能够以完全无人照管的方式运行。这样便可以轻松自动完成 Guest 安装。**virt-install** 是虚拟机管理器使用的默认安装工具。

5.2 虚拟化 GUI 工具

SUSE Linux Enterprise Server 提供了以下基于 libvirt 的图形工具。所有工具由带有相应工具名称的软件包提供。

虚拟机管理器 (软件包: virt-manager)

虚拟机管理器是用于管理 VM Guest 的桌面工具。此工具提供控制现有计算机生命周期（启动/关机、暂停/继续、保存/恢复）以及创建新 VM Guest 的功能。它可用于管理各种类型的储存设备和虚拟网络。使用它可以通过内置 VNC 查看器访问 VM Guest 的图形控制台，以及查看性能统计信息。**virt-manager** 支持连接到本地 libvirtd 来管理本地 VM 主机服务器，或连接到远程 libvirtd 来管理远程 VM 主机服务器。



要启动虚拟机管理器，请在命令提示符处输入 **virt-manager**。



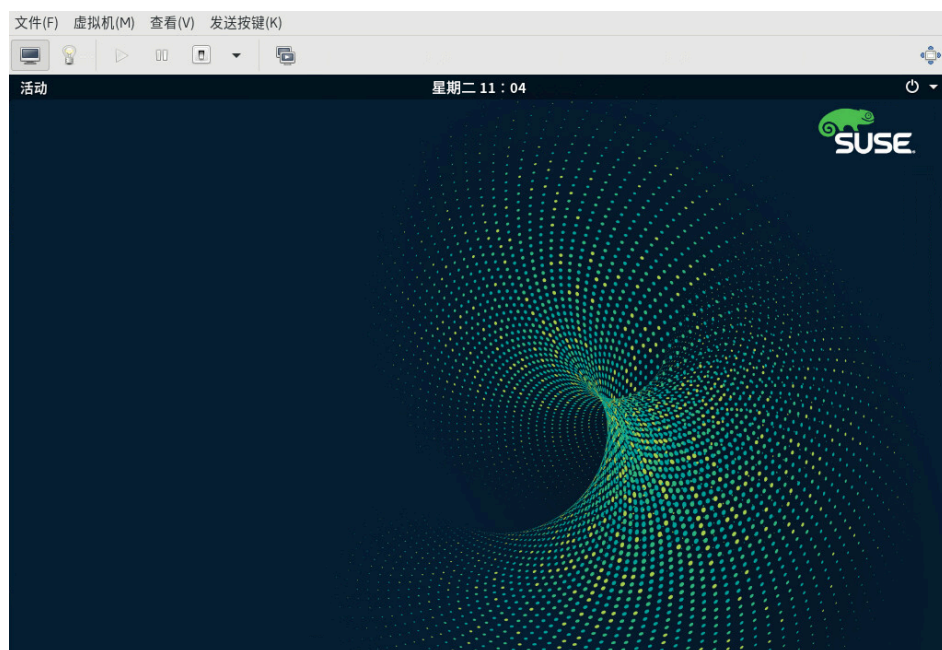
注意

要禁用通过 spice 对 VM Guest 自动进行 USB 设备重定向的功能，请结合 **--spice-disable-auto-usbredir** 参数启动 **virt-manager**，或运行以下命令来永久更改默认行为：

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

virt-viewer (软件包: **virt-viewer**)

VM Guest 图形控制器的查看器。它使用 SPICE（默认已在 VM Guest 上配置）或 VNC 协议，并支持 TLS 和 x509 证书。可按名称、ID 或 UUID 访问 VM Guest。如果 Guest 尚未运行，可以告知该查看器先等待 Guest 启动，然后再尝试连接到控制台。**virt-viewer** 默认未安装，安装软件包 **virt-viewer** 后即可使用它。



注意

要禁用通过 spice 对 VM Guest 自动进行 USB 设备重定向的功能，请使用 `--spice-usbredir-auto-redirect-filter=''` 参数添加一个空过滤器。

yast2 vm（软件包：yast2-vm）

一个 YaST 模块，可简化虚拟化工具的安装并可设置网桥：

选择要安装的虚拟机管理程序

服务器：运行虚拟机管理程序的最小系统

工具：配置、管理和监视虚拟机

禁止选择的复选框意味着 Hypervisor 项目已被安装

Xen 管理程序

☐ Xen 服务器

☐ Xen 工具

KVM 管理程序

☐ KVM 服务器

☐ KVM 工具

libvirt LXC 容器

☐ libvirt LXC 守护进程

取消(C)

接受(A)

6 安装虚拟化组件

根据安装范围，可能不会在您的系统上安装任何虚拟化工具。使用 YaST 模块 [虚拟化 > 安装超级管理程序和工具](#) 配置超级管理程序时，会自动安装这些工具。如果 YaST 中未提供此模块，请安装软件包 [yast2-vm](#)。

6.1 安装 KVM

要安装 KVM 和 KVM 工具，请执行以下操作：

1. 确认是否已安装 [yast2-vm](#) 包是否已安装。此软件包是 YaST 的配置工具，可以简化虚拟化超级管理程序的安装过程。
2. 启动 YaST 并选择 [虚拟化 > 安装超级管理程序和工具](#)。
3. 选择 KVM 服务器以安装最少量的 QEMU 工具。如果还需要基于 [libvirt](#) 的管理堆栈，请选择 KVM 工具。使用接受确认。
4. 要为 VM Guest 启用正常网络功能，建议使用网桥。YaST 会建议自动在 VM 主机服务器上配置网桥。如果您同意执行此操作，请选择是，否则请选择否。
5. 安装完成后，您便可以开始设置 VM Guest。不需要重引导 VM 主机服务器。

6.2 安装 Xen

要安装 Xen 和 Xen 工具，请执行以下操作：

1. 启动 YaST 并选择 [虚拟化 > 安装超级管理程序和工具](#)。
2. 选择 Xen 服务器以安装最少量的 Xen 工具。如果还需要基于 [libvirt](#) 的管理堆栈，请选择 Xen 工具。使用接受确认。
3. 要为 VM Guest 启用正常网络功能，建议使用网桥。YaST 会建议自动在 VM 主机服务器上配置网桥。如果您同意执行此操作，请选择是，否则请选择否。

4. 安装完成后，您需使用 Xen 内核重引导计算机。



提示：默认引导内核

如果一切按预期进行，请使用 YaST 更改默认引导内核，并将支持 Xen 的内核设置为默认内核。有关更改默认内核的详细信息，请参见《管理指南》，第 14 章“引导加载程序 GRUB 2”，第 14.3 节“使用 YaST 配置引导加载程序”。

6.3 安装容器

要安装容器，请执行以下操作：

1. 启动 YaST 并选择虚拟化 › 安装超级管理程序和工具。
2. 选择 libvirt lxc 守护程序并单击接受以确认。

6.4 模式

可以使用 Zypper 和模式来安装虚拟化软件包。运行 `zypper in -t pattern PATTERN` 命令。可用模式包括：

KVM

- `kvm_server`：为 KVM VM 主机服务器设置用于进行管理的 QEMU 工具
- `kvm_tools`：安装用于管理和监视 VM Guest 的 `libvirt` 工具

Xen

- `xen_server`：为 Xen VM 主机服务器设置用于进行管理的 Xen 工具
- `xen_tools`：安装用于管理和监视 VM Guest 的 `libvirt` 工具

容器

容器没有模式；请安装 `libvirt-daemon-lxc` 软件包。

6.5 安装 UEFI 支持

UEFI 支持由 OVMF（开放虚拟机固件）提供。要启用 UEFI 引导，请先安装 `qemu-ovmf-x86_64` 或 `qemu-uefi-aarch64` 软件包。

系统会自动选择虚拟机使用的固件。自动选择是根据 `qemu-ovmf-ARCH` 软件包中的 *.json 文件做出的。`libvirt` QEMU 驱动程序会在装载时分析这些文件，因此知道各种固件的功能。然后，当用户选择固件类型以及任何所需功能（例如，安全引导支持）时，`libvirt` 能够找到符合用户要求的固件。

例如，要指定支持安全引导的 EFI，请使用以下配置：

```
<os firmware='efi'>
  <loader secure='yes' />
</os>
```

软件包 `qemu-ovmf-ARCH` 中包含以下文件：

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

`*-code.bin` 文件是 UEFI 固件文件。`*-vars.bin` 文件是对应的变量储存映像，可用作每个 VM 的非易失性储存模板。首次创建 VM 时，`libvirt` 会将指定的 `vars` 模板复制到每个 VM 路径的 `/var/lib/libvirt/qemu/nvram/` 下。名称中不包含 `code` 或 `vars` 的文件可用作单个 UEFI 映像。它们没有太大的作用，因为每次经过 VM 关开机周期后，UEFI 变量都不会保存。

`*-ms*.bin` 文件包含存放在实际硬件上的 Microsoft 密钥。因此，在 `libvirt` 中它们已配置为默认设置。同样，`*-suse*.bin` 文件包含预安装的 SUSE 密钥。还有一组不包含预安装密钥的文件。

有关细节，请参见[使用 UEFI 和安全引导和http://www.linux-kvm.org/downloads/lersek/ovmf-whitepaper-c770f8c.txt](http://www.linux-kvm.org/downloads/lersek/ovmf-whitepaper-c770f8c.txt)。

6.6 启用 KVM 中的嵌套式虚拟化支持

！ 重要：技术预览

KVM 的嵌套式虚拟化仍为技术预览版。此版本是出于测试目的提供的，我们不提供相关支持。

嵌套式 Guest 是 KVM Guest 中运行的 KVM Guest。在描述嵌套式 Guest 时，我们会用到以下虚拟化层：

L0

运行 KVM 的裸机主机。

L1

在 L0 上运行的虚拟机。由于它可以在另一个 KVM 上运行，因此称为 Guest 超级管理程序。

L2

在 L1 上运行的虚拟机。称为嵌套式 Guest。

嵌套式虚拟化具有诸多优势。它可在以下场景中为您提供便利：

- 在云环境中直接使用所选的超级管理程序管理您自己的虚拟机。
- 将超级管理程序及其 Guest 虚拟机作为单个实体进行实时迁移。
- 使用嵌套式虚拟化进行软件开发和测试。

要暂时启用嵌套功能，请去除该模块，然后使用 `nested` KVM 模块参数重新装载该模块：

- 对于 Intel CPU，请运行：

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

- 对于 AMD CPU，请运行：

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

要永久启用嵌套功能，请根据您的 CPU，在 /etc/modprobe.d/kvm_*.conf 文件中启用 nested KVM 模块参数：

- 对于 Intel CPU，请编辑 /etc/modprobe.d/kvm_intel.conf 并添加下面一行：

```
options kvm_intel nested=Y
```

- 对于 AMD CPU，请编辑 /etc/modprobe.d/kvm_amd.conf 并添加下面一行：

```
options kvm_amd nested=Y
```

如果 L0 主机能够嵌套，则您可以通过以下方式之一启动 L1 Guest：

- 使用 -cpu host QEMU 命令行选项。
- 将 vmx（对于 Intel CPU）或 svm（对于 AMD CPU）CPU 功能添加到 -cpu QEMU 命令行选项，用于启用虚拟 CPU 的虚拟化。

7 支持的主机、Guest 和功能

《Release Notes》（发行说明）(<https://www.suse.com/releasenotes/>) 中概述了支持的体系结构以及 Xen 和 KVM 的虚拟化限制。

7.1 主机环境（超级管理程序）

本节列出了在各种虚拟化主机（超级管理程序）上作为 Guest 运行的 SUSE Linux Enterprise Server 15 SP2 的支持状态。

支持以下 SUSE 主机环境 (XEN + KVM)：

- SLES 11 SP4
- SLES 12 SP1、SP2、SP3、SP4、SP5
- SLES 15 SP0、SP1、SP2

支持以下第三方主机环境：

- VMware ESXi 6.5、6.7
- Microsoft Windows 2008 R2 SP1+、2012+、2012 R2+、2016、2019
- Citrix XenServer 7.0、7.1、8.0
- Oracle VM 3.4

支持级别如下：

- 对于 Guest 和主机，SUSE 主机操作系统的支持级别均为全面 L3。
- 对于 Guest，第三方主机环境的支持级别为全面 L3；对于主机，需要主机供应商的协作与支持。



注意：对第三方虚拟化主机的支持

仅当虚拟化主机和 Guest 都使用 SUSE 产品时，SUSE 才提供全面 L3 支持。

如果虚拟化主机使用第三方软件，则 SUSE 只能保证为 Guest 提供 L3 支持。对于主机，需要供应商的协作。

7.2 Guest 环境

本节列出了在 SUSE Linux Enterprise Server 15 SP2 上虚拟化的各种 Guest 操作系统的支持状态。所有全虚拟化（在下表中以“FV”表示）和半虚拟化（在下表中以“PV”表示）的 Guest 操作系统均受支持，但存在两种例外情况：对于 Windows，仅支持全虚拟化状态，对于 NetWare 操作系统，仅在 Xen 上支持半虚拟化状态。除非另有说明，否则所有 32 位和 64 位版本的 Guest 操作系统均受支持（请参见“NetWare”）。

以下 GUEST 操作系统受到全面支持 (L3)：

- SLES 10 SP4
- SLES 11 SP3、SP4
- SLES 12 SP0、SP1、SP2、SP3、SP4、SP5
- SLES 15 SP0、SP1、SP2
- OES 11 SP2、2015、2015 SP1、2018、2018 SP1、2018 SP2
- Netware 6.5 SP8（仅限 32 位）
- Windows Server 2008 SP2+、2008 R2 SP1+、2012+、2012 R2+、2016、2019

以技术预览（L2，在合理的情况下提供修复）的形式支持以下 GUEST 操作系统：

- SLED 15 SP1

如果客户购买了扩展支持，则以下 RED HAT GUEST 操作系统受到全面支持 (L3)，否则会按照尽力而为的原则为其提供支持（L2，在合理的情况下提供修复）：

- RHEL 5.11+、6.9+、7.7+、8.0+

按照尽力而为的原则为以下 GUEST 操作系统提供支持（L2，在合理的情况下提供修复）：

- Windows 8+、8.1+、10+

7.2.1 半虚拟化驱动程序提供

为了提升 Guest 操作系统的性能，我们将会提供半虚拟化驱动程序（如果有）。尽管这些驱动程序不是必需的，但我们强烈建议使用。半虚拟化驱动程序的提供方式如下：

SUSE Linux Enterprise Server 12/12 SP1/12 SP2

包含在内核中

SUSE Linux Enterprise Server 11/11 SP1/11 SP2/11 SP3/11 SP4

包含在内核中

SUSE Linux Enterprise Server 10SP4

包含在内核中

RedHat

在 RedHat Enterprise Linux 5.4 和更高版本中提供

Windows

SUSE 开发了适用于 Windows 的基于 virtio 的驱动程序，这些驱动程序包含在虚拟机驱动程序包 (VMDP) 中。有关更多信息，请参见<https://www.suse.com/products/vmdriverpack/>。

7.3 KVM 硬件要求

目前，SUSE 支持 AMD64/Intel 64 和 Arm AArch64 主机以及 IBM Z 上的 KVM 全虚拟化。

- 在 AMD64/Intel 64 体系结构上，KVM 是围绕 AMD* (AMD-V) 和 Intel* (VT-x) CPU 中包含的硬件虚拟化功能设计的。它支持芯片组和 PCI 设备的虚拟化功能，例如 I/O 内存映射单元 (IOMMU) 和单根 I/O 虚拟化 (SR-IOV)。您可以使用以下命令测试您的 CPU 是否支持硬件虚拟化：

```
tux > egrep '(vmx|svm)' /proc/cpuinfo
```

如果此命令未返回任何输出，则表示您的处理器不支持硬件虚拟化，或者已在 BIOS 或固件中禁用此功能。

以下网站指出了支持硬件虚拟化的 AMD64/Intel 64 处理器：<http://ark.intel.com/Products/VirtualizationTechnology>（针对 Intel CPU），以及 <http://products.amd.com/>（针对 AMD CPU）。

- 在 Arm 体系结构上，最初是从 Arm Cortex-A15 开始向 Armv7-A 处理器添加虚拟化支持的，其中包括 Cortex-A7 和 Cortex-A17。Armv8-A 处理器包含虚拟化支持。



注意：不装载 KVM 内核模块

仅当 CPU 硬件虚拟化功能可用时，才会装载 KVM 内核模块。

VM 主机服务器的一般性最低硬件要求与《部署指南》，第 2 章“在 AMD64 和 Intel 64 上安装”，第 2.1 节“硬件要求”中概述的相同。不过，对于每个虚拟化的 Guest 都需要提供额外的 RAM。此额外 RAM 量应至少与物理安装所需的 RAM 量相同。另外，强烈建议为每个运行中的 Guest 至少配备一个处理器核心或超线程。

7.4 功能支持

7.4.1 主机 (Dom0)

表 7.1：功能支持 — 主机 (Dom0)

功能	Xen
网络和块设备热插拔	是
物理 CPU 热插拔	否
虚拟 CPU 热插拔	是
虚拟 CPU 固定	是
虚拟 CPU 限制	是
Intel* VT-x2: FlexPriority、FlexMigrate（迁移限制适用于不同的 CPU 体系结构）	是
Intel* VT-d2（具有中断过滤和排队失效的 DMA 重新映射）	是
AMD* IOMMU（具有 Guest 到主机物理地址转换的 I/O 页表）	是



注意：不支持在运行时添加或去除物理 CPU

不支持在运行时添加或去除物理 CPU，但可以添加或去除每个 VM Guest 的虚拟 CPU。

7.4.2 半虚拟化 Guest

表 7.2：功能支持 — 半虚拟化 GUEST

功能	Xen
虚拟网络和虚拟块设备热插拔	是
虚拟 CPU 热插拔	是
虚拟 CPU 过量分配	是
动态虚拟内存大小调整	是
VM 保存和恢复	是
VM 实时迁移	是，在具有类似资源的类似虚拟主机系统之间
使用 GDB 进行高级调试	是
对 VM 可见的 Dom0 度量	是
内存气球	是
PCI 直通	是（不包括 Netware Guest）

对于实时迁移，源体系结构和目标体系结构均需要匹配；即处理器（AMD* 或 Intel*）必须相同。除非使用了 CPU ID 掩码（例如，使用 Intel FlexMigration），否则目标的处理器修订版应该与源相同或者比源更新。如果在不同系统之间移动 VM，那么这些规则适用于每一次移动。为了避免优化的代码在运行时或应用程序启动期间失败，源 CPU 和目标 CPU 需要公开相同的处理器扩展。Xen 透明地向 VM 公开物理 CPU 扩展。总而言之，Guest 可以是 32 位或 64 位，但 VHS 必须相同。



注意：Intel FlexMigration

对于支持 Intel FlexMigration 的计算机，CPU-ID 掩码和错误引发可以提高跨 CPU 迁移的灵活性。

7.4.3 全虚拟化 Guest

表 7.3：功能支持 — 全虚拟化 GUEST

功能	Xen	KVM
虚拟网络和虚拟块设备热插拔	是	是
虚拟 CPU 热插拔	否	否
虚拟 CPU 过量分配	是	是
动态虚拟内存大小调整	是	是
VM 保存和恢复	是	是
VM 实时迁移	是，在具有类似资源的类似虚拟主机系统之间（即，从 32 位迁移到 32 位，或从 64 位迁移到 64 位）	是
VM 快照	是	是
使用 GDB 进行高级调试	是	是
对 VM 可见的 Dom0 度量	是	是
PCI 直通	是	是



注意：Windows Guest

仅当使用的是 PV 驱动程序 (VMDP) 时，才支持在 Xen 和 KVM 中热插拔虚拟网络和虚拟块设备，以及收缩、恢复动态虚拟内存及调整其大小。

对于 KVM，有关支持的限制、功能、建议的设置和方案的详细说明以及其他有用信息均在 [kvm-supported.txt](#) 中提供。此文件是 KVM 软件包的一部分，可在 [/usr/share/doc/packages/kvm](#) 中找到。

II 使用 libvirt 管理虚拟机

- 8 启动和停止 libvirtd 31
- 9 Guest 安装 33
- 10 基本 VM Guest 管理 40
- 11 连接和授权 66
- 12 管理储存 87
- 13 管理网络 106
- 14 使用虚拟机管理器配置虚拟机 122
- 15 使用 **virsh** 配置虚拟机 143
- 16 使用 Vagrant 管理虚拟机 172

8 启动和停止 libvirtd

虚拟化解决方案（KVM、Xen、LXC）与 libvirt API 之间的通讯由 libvirtd 守护程序来管理。此守护程序需在 VM 主机服务器上运行。可能在远程计算机上运行的 libvirt 客户端应用程序（例如 virt-manager）会与 VM 主机服务器上运行的 libvirtd 通讯。libvirtd 使用本机超级管理程序 API 来处理请求。使用以下命令可以启动和停止 libvirtd 或检查其状态：

```
tux > sudo systemctl start libvirtd

tux > sudo systemctl status libvirtd
libvirtd.service - Virtualization daemon
Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled)
Active: active (running) since Mon 2014-05-12 08:49:40 EDT; 2s ago
[...]

tux > sudo systemctl stop libvirtd

tux > sudo systemctl status libvirtd
[...]
Active: inactive (dead) since Mon 2014-05-12 08:51:11 EDT; 4s ago
[...]
```

要在系统引导时自动启动 libvirtd，请使用 YaST 服务管理器模块或输入以下命令将其激活：

```
tux > sudo systemctl enable libvirtd
```

❗ 重要：有冲突的服务：libvirtd 和 xendomains

如果 libvirtd 无法启动，请检查是否装载了 xendomains 服务：

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

如果该命令返回 active，您需要停止 xendomains，然后才可以启动 libvirtd 守护程序。如果您希望在重引导后也要启动 libvirtd，另外还需禁止 xendomains 自动启动。禁用该服务：

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

xendomains 和 libvirtd 提供的是相同的服务，如果同时使用，可能会互相干扰。例如，xendomains 可能会尝试启动已由 libvirtd 启动的 domU。

9 Guest 安装

VM Guest 由一个包含操作系统和数据文件的映像以及一个描述 VM Guest 虚拟硬件资源的配置文件构成。VM Guest 托管在 VM 主机服务器上并受其控制。本节提供有关安装 VM Guest 的概括说明。有关支持的 VM Guest 列表，请参见第 7 章“支持的主机、Guest 和功能”。

与运行操作系统需要满足的要求相比，虚拟机几乎没有什么要求。如果操作系统未根据虚拟机主机环境进行优化，它将只能以全虚拟化模式在硬件辅助虚拟化计算机硬件上运行，并需要装载特定的设备驱动程序。提供给 VM Guest 的硬件取决于主机的配置。

您应该了解与在多个虚拟机上运行单个已许可操作系统副本相关的任何许可问题。有关详细信息，请查阅操作系统许可协议。

9.1 基于 GUI 的 Guest 安装

新建 VM 向导将帮助您完成创建虚拟机和安装其操作系统所需执行的步骤。可通过两种方式启动该向导：在虚拟机管理器中，单击创建新虚拟机或选择文件 > 新建虚拟机。或者启动 YaST 并选择虚拟化 > 创建 Xen 和 KVM 的虚拟机。

1. 从 YaST 或虚拟机管理器中启动新建 VM 向导。
2. 选择安装源 — 本地可用的媒体或网络安装源。如果您要从现有映像安装 VM Guest，请选择导入现有磁盘映像。
在运行 Xen 超级管理程序的 VM 主机服务器上，您可以选择是要安装半虚拟化 Guest 还是全虚拟化 Guest。您可以在体系结构选项下选择相应的选项。根据此项选择，并非所有安装选项均可用。
3. 根据在上一步中所做的选择，您需要提供以下数据：

本地安装媒体（ISO 映像或 CDROM）

在 VM 主机服务器上指定包含安装数据的 ISO 映像的路径。如果该映像是作为 libvirt 储存池中的卷提供的，您也可以使用浏览来选择。有关更多信息，请参见第 12 章“管理储存”。

或者，选择已插入到 VM 主机服务器光驱中的物理 CD-ROM 或 DVD。

网络安装 (HTTP、FTP 或 NFS)

提供指向安装源的 URL。有效的 URL 前缀包括 [ftp://](#)、[http://](#)、[https://](#) 和 [nfs://](#) 等。

在 URL 选项下，提供自动安装文件（例如 AutoYaST 或 Kickstart）的路径以及内核参数。提供 URL 后，应该就会自动正确检测到操作系统。如果情况并非如此，请取消选择基于安装媒体自动检测操作系统，并手动选择操作系统类型和版本。

网络引导 (PXE)

通过 PXE 引导时，您只需提供操作系统类型和版本。

导入现有磁盘映像

要从现有映像安装 VM Guest，您需要在 VM 主机服务器上指定该映像的路径。如果该映像是作为 libvirt 储存池中的卷提供的，您也可以使用浏览来选择。有关更多信息，请参见第 12 章 “管理储存”。

4. 选择新虚拟机的内存大小和 CPU 数量。
5. 如果在第一步中选择了导入现有映像，则会省略此步骤。
设置 VM Guest 的虚拟硬盘。创建新磁盘映像，或者从储存池中选择一个现有的磁盘映像（有关详细信息，请参见第 12 章 “管理储存”）。如果您选择创建磁盘，将会创建一个 [qcow2](#) 映像。默认情况下，该映像储存在 [/var/lib/libvirt/images](#) 下。
设置磁盘是可选操作。例如，如果您正在直接从 CD 或 DVD 运行实时系统，可以通过停用为此虚拟机启用储存来省略此步骤。
6. 在向导的最后一个屏幕上指定虚拟机的名称。如果您希望能够查看和更改虚拟化硬件选择，请激活在安装之前自定义配置。网络选择下提供了用于指定网络设备的选项。
单击完成。
7. （可选）如果您在上一步中保留了默认设置，安装现在就会开始。如果您选择了在安装之前自定义配置，一个 VM Guest 配置对话框将会打开。有关配置 VM Guest 的详细信息，请参见第 14 章 “使用虚拟机管理器配置虚拟机”。
完成配置后，单击开始安装。



提示：将组合键传递给虚拟机

安装将在一个虚拟机管理器控制台窗口中开始。某些组合键（例如 **Ctrl - Alt - F1**）会被 VM 主机服务器接受，但不会传递给虚拟机。虚拟机管理器提供“粘滞键”功能来绕过 VM 主机服务器。按 **Ctrl**、**Alt** 或 **Shift** 三次使该键成为粘滞键，然后按组合键中剩余的键便可将组合键传递给虚拟机。

例如，要将 **Ctrl - Alt - F2** 传递给 Linux 虚拟机，请按 **Ctrl** 三次，然后按 **Alt - F2**。也可以按 **Alt** 三次，然后按 **Ctrl - F2**。

在安装 VM Guest 期间以及安装之后，都可以在虚拟机管理器中使用粘滞键功能。

9.2 在命令行中使用 **virt-install** 安装

virt-install 是个命令行工具，可帮助您使用 **libvirt** 库创建新虚拟机。如果您无法使用图形用户界面，或需要自动化虚拟机创建过程，此工具十分有用。

virt-install 是个复杂的脚本，其中包含大量命令行开关。下面是必需的开关。有关详细信息，请参见 **virt-install** (1) 手册页。

一般选项

- **--name VM_GUEST_NAME**：指定新虚拟机的名称。该名称必须在同一连接上超级管理程序已知的所有 Guest 中保持唯一。该名称用于创建和命名 Guest 的配置文件，您可以通过 **virsh** 使用该名称来访问 Guest。名称可包含字母数字和 **_-.:+** 字符。
- **--memory REQUIRED_MEMORY**：以 MB 为单位指定分配给新虚拟机的内存量。
- **--vcpus NUMBER_OF_CPUS**：指定虚拟 CPU 数量。要获得最佳性能，虚拟处理器数量应小于或等于物理处理器数量。

虚拟化类型

- `--paravirt`：安装半虚拟化 Guest。如果 VM 主机服务器支持半虚拟化和全虚拟化，这就是默认设置。
- `--hvm`：安装全虚拟化 Guest。
- `--virt-type HYPERVISOR`：指定超级管理程序。支持的值为 `kvm`、`xen` 或 `lxc`。

Guest 储存

指定 `--disk`、`--filesystem` 或 `--nodisks` 作为新虚拟机的储存类型。例如，`--disk size=10` 会在超级管理程序的默认映像位置创建 10 GB 磁盘，并将此磁盘用于 VM Guest。`--filesystem /export/path/on/vmhost` 指定 VM 主机服务器上要导出到 Guest 的目录。`--nodisks` 安装不带本地储存的 VM Guest（适合使用实时 CD 的情形）。

安装方法

使用 `--location`、`--cdrom`、`--pxe`、`--import` 或 `--boot` 指定安装方法。

访问安装

使用 `--graphics VALUE` 选项指定如何访问安装。SUSE Linux Enterprise Server 支持值 `vnc` 或 `none`。

如果使用 VNC，`virt-install` 将尝试启动 `virt-viewer`。如果 `virt-viewer` 未安装或无法运行，请使用您偏好的查看器手动连接到 VM Guest。要显式阻止 `virt-install` 启动查看器，请使用 `--noautoconsole`。要定义用于访问 VNC 会话的口令，请使用以下语法：`--graphics vnc,password=PASSWORD`。

如果您使用 `--graphics none`，可以通过操作系统支持的服务（例如 SSH 或 VNC）访问 VM Guest。请参见操作系统安装手册了解如何在安装系统中设置这些服务。

传递内核和 Initrd 文件

可以直接指定安装程序的内核和 Initrd，例如，指定来自网络来源的内核和 Initrd。要设置网络来源，请参见《部署指南》，第 16 章“设置网络安装源”，第 16.4 节“手动设置 HTTP 储存库”。

要传递其他引导参数，请使用 `--extra-args` 选项。此选项可用于指定网络配置。有关详细信息，请参见《部署指南》，第 7 章“引导参数”。

例 9.1：从 HTTP 服务器装载内核和 INITRD

```
root # virt-install --location "http://example.tld/REPOSITORY/DVD1/" \
--extra-args="textmode=1" --name "SLES15" --memory 2048 --virt-type kvm \
--connect qemu:///system --disk size=10 --graphics vnc \
--network network=vnet_nated
```

启用控制台

默认不会对使用 **virt-install** 安装的新虚拟机启用控制台。要启用控制台，请按下面的示例所示使用 `--extra-args="console=ttyS0 textmode=1"`：

```
tux > 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
```

安装完成后，VM 映像中的 `/etc/default/grub` 文件将会更新，在 `GRUB_CMDLINE_LINUX_DEFAULT` 行中包含 `console=ttyS0` 选项。

使用 UEFI 和安全引导

按照第 6.5 节“安装 UEFI 支持”中所述安装 OVMF。然后将 `--boot uefi` 选项添加到 **virt-install** 命令。

设置使用 OVMF 的新 VM 时，将自动使用安全引导。要使用特定的固件，请使用 `--boot loader=/usr/share/qemu/ovmf-VERSION.bin`。请将 `VERSION` 替换为所需的文件。

例 9.2：virt-install 命令行示例

下面的命令行示例会创建带有 virtio 加速磁盘和网卡的新 SUSE Linux Enterprise Desktop 12 虚拟机。它将创建新的 10 GB qcow2 磁盘映像作为储存空间，源安装媒体为主机 CD-ROM 驱动器。此命令行将使用 VNC 图形，并自动启动图形客户端。

KVM

```
tux > virt-install --connect qemu:///system --virt-type kvm --name
sled12 \
--memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc \
--os-variant sled12
```

```
tux > virt-install --connect xen:// --virt-type xen --name sled12 \
--memory 1024 --disk size=10 --cdrom /dev/cdrom --graphics vnc \
--os-variant sled12
```

9.3 高级 Guest 安装方案

本节提供有关超出了正常安装范围的操作（例如使用内存气球和安装附加产品）的说明。

9.3.1 在 Windows Guest 上使用内存气球

内存气球是在运行时更改 VM Guest 所用内存量的方法。KVM 和 Xen 超级管理程序都提供此方法，但需要 Guest 也支持此方法。

基于 openSUSE 和 SLE 的 Guest 支持内存气球，而 Windows Guest 需要通过[虚拟机驱动程序包 \(VMDP\)](https://www.suse.com/products/vmdriverpack/) (<https://www.suse.com/products/vmdriverpack/>) 来提供气球技术。要使设置的最大内存大于为 Windows Guest 配置的初始内存，请执行以下步骤：

1. 安装最大内存等于或小于初始值的 Windows Guest。
2. 在 Windows Guest 中安装虚拟机驱动程序包，以提供所需的驱动程序。
3. 关闭 Windows Guest。
4. 将 Windows Guest 的最大内存重设置为所需值。
5. 再次启动 Windows Guest。

9.3.2 在安装中包含附加产品

某些操作系统（例如 SUSE Linux Enterprise Server）允许在安装过程中包含附加产品。如果附加产品安装源是通过 SUSE Customer Center 提供的，则无需进行特殊的 VM Guest 配置。如果安装源是通过 CD/DVD 或 ISO 映像提供的，则需要向 VM Guest 安装系统提供标准安装媒体映像和附加产品的映像。

如果您使用的是基于 GUI 的安装，请在向导的最后一步选择在安装之前自定义配置，并通过添加硬件 > 储存添加附加产品 ISO 映像。指定映像的路径，并将设备类型设置为 CD-ROM。

如果您是从命令行安装的，则需要使用 `--disk` 参数而不是 `--cdrom` 来设置虚拟 CD/DVD 驱动器。将使用第一个指定的设备进行引导。下面的示例会将 SUSE Enterprise Storage 扩展随 SUSE Linux Enterprise Server 15 一并安装：

```
tux > virt-install --name sles15+storage --memory 2048 --disk size=10 \  
--disk /path/to/SLE-15-SP2-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 sles15
```

10 基本 VM Guest 管理

使用虚拟机管理器图形应用程序或者在命令行上使用 `virsh` 可以完成大部分管理任务，例如启动或停止 VM Guest。而要通过 VNC 连接到图形控制台，就只能从图形用户界面进行。



注意：管理远程 VM 主机服务器上的 VM Guest

如果 VM 主机服务器上启动了虚拟机管理器、`virsh` 和 `virt-viewer` 这些 `libvirt` 工具，则可以使用它们来管理主机上的 VM Guest。不过，您也可以管理远程 VM 主机服务器上的 VM Guest。这需要在主机上为 `libvirt` 配置远程访问权限。有关说明，请参见第 11 章“连接和授权”。

要使用虚拟机管理器连接到此类远程主机，需要按照第 11.2.2 节“使用虚拟机管理器管理连接”中所述设置连接。如果通过 `virsh` 或 `virt-viewer` 连接到远程主机，需要使用参数 `-c` 指定连接 URI（例如，`virsh -c qemu+tls://saturn.example.com/system` 或 `virsh -c xen+ssh://`）。连接 URI 的格式取决于连接类型和超级管理程序 — 有关细节，请参见第 11.2 节“连接到 VM 主机服务器”。

本章列出的所有示例都不包含连接 URI。

10.1 列出 VM Guest

VM Guest 列表显示 VM 主机服务器上由 `libvirt` 管理的所有 VM Guest。

10.1.1 使用虚拟机管理器列出 VM Guest

虚拟机管理器的主窗口会列出它所连接的每台 VM 主机服务器的所有 VM Guest。每个 VM Guest 项包含该计算机的名称及状态（正在运行、已暂停或已关闭），这些信息以图标、文本和 CPU 使用率条的形式显示。

10.1.2 使用 **virsh** 列出 VM Guest

使用 **virsh list** 命令可获取 VM Guest 的列表：

列出所有正在运行的 Guest

```
tux > virsh list
```

列出所有正在运行的 Guest 以及非活动的 Guest

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

有关详细信息和其他选项，请参见 **virsh help list** 或 **man 1 virsh**。

10.2 通过控制台访问 VM Guest

可以通过 VNC 连接（图形控制台）或串行控制台（如果受 Guest 操作系统的支持）访问 VM Guest。

10.2.1 打开图形控制台

打开与 VM Guest 连接的图形控制台可与该计算机交互，就如同通过 VNC 连接与物理主机交互一样。如果访问 VNC 服务器需要身份验证，系统会提示您输入用户名（如果适用）和口令。

当您单击进入 VNC 控制台时，光标将被“夺取”，不能再在控制台外部使用。要释放光标，请按 **Alt - Ctrl**。



提示：无缝（绝对）光标移动

为防止控制台夺取光标，同时为了启用无缝光标移动，请向 VM Guest 添加绘图板输入设备。有关更多信息，请参见第 14.5 节“输入设备”。

某些组合键（例如 **Ctrl - Alt - Del**）由主机解释，不会传递给 VM Guest。要将此类组合键传递给 VM Guest，请在 VNC 窗口中打开发送键菜单，然后选择所需的组合键项。仅当使用虚拟机管理器和 **virt-viewer** 时，发送键菜单才可用。借助虚拟机管理器，可以按照提示：将组合键传递给虚拟机中所述改用“粘滞键”功能。



注意：支持的 VNC 查看器

理论上而言，所有 VNC 查看器都可连接到 VM Guest 的控制台。但如果您是使用 SASL 身份验证和/或 TLS/SSL 连接来访问 Guest 的，那么您的选择就比较有限。**tightvnc** 或 **tigervnc** 等常见 VNC 查看器既不支持 SASL 身份验证，也不支持 TLS/SSL。唯一可替代虚拟机管理器和 **virt-viewer** 的工具是 Remmina（请参见《管理指南》，第 10 章“使用 VNC 的远程图形会话”，第 10.2 节“Remmina：远程桌面客户端”）。

10.2.1.1 使用虚拟机管理器打开图形控制台

1. 在虚拟机管理器中，右键单击某个 VM Guest 项。
2. 从弹出菜单中选择打开。

10.2.1.2 使用 **virt-viewer** 打开图形控制台

virt-viewer 是一个简单的 VNC 查看器，其中添加了用于显示 VM Guest 控制台的功能。例如，可以“wait”模式启动该查看器，在此情况下，它会先等待 VM Guest 启动，然后再建立连接。它还支持自动重新连接到重引导的 VM Guest。

virt-viewer 按名称、ID 或 UUID 对 VM Guest 进行寻址。使用 **virsh list --all** 可获取这些数据。

要连接到正在运行或已暂停的 Guest，请使用 ID、UUID 或名称。已关闭的 VM Guest 没有 ID — 您只能按 UUID 或名称与其建立连接。

连接到 ID 为 8 的 Guest

```
tux > virt-viewer 8
```

连接到名为 **sles12** 的非活动 Guest；Guest 启动后，连接窗口就会打开

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

如果使用 **--wait** 选项，即使 VM Guest 此刻未运行，也会保持连接。当 Guest 启动时，查看器即会启动。

有关详细信息，请参见 `virt-viewer --help` 或 `man 1 virt-viewer`。



注意：通过 SSH 建立远程连接时输入的口令

使用 `virt-viewer` 通过 SSH 来与远程主机建立连接时，需要输入 SSH 口令两次。第一次用于向 `libvirt` 进行身份验证，第二次用于向 VNC 服务器进行身份验证。第二个口令需要在启动 `virt-viewer` 的命令行上提供。

10.2.2 打开串行控制台

要访问虚拟机的图形控制台，需要在访问 VM Guest 的客户端上提供一个图形环境。或者，也可通过串行控制台和 `virsh` 在外壳中访问使用 `libvirt` 管理的虚拟机。要打开与名为 “sles12” 的 VM Guest 连接的串行控制台，请运行以下命令：

```
tux > virsh console sles12
```

`virsh console` 接受两个可选标志：`--safe` 确保以独占方式访问控制台，`--force` 在连接之前断开与所有现有会话的连接。这两个功能需受 Guest 操作系统的支持。

Guest 操作系统必须支持串行控制台访问，并且该操作系统也受到适当的支持，才能通过串行控制台连接到 VM Guest。有关详细信息，请参见 Guest 操作系统手册。



提示：为 SUSE Linux Enterprise Guest 和 openSUSE Guest 启用串行控制台访问

SUSE Linux Enterprise 和 openSUSE 中默认会禁用串行控制台访问。要启用它，请执行下列步骤：

SLES 12 和 Up/openSUSE

启动 YaST 引导加载程序模块并切换到内核参数选项卡。将 `console=ttyS0` 添加到字段可选内核命令行参数。

SLES 11

启动 YaST 引导加载程序模块，并选择要为其激活串行控制台访问的引导项。选择编辑，并将 `console=ttyS0` 添加到字段可选内核命令行参数。此外，请编辑 `/etc/inittab` 并取消注释包含以下内容的行：

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

10.3 更改 VM Guest 的状态：启动、停止、暂停

可以使用虚拟机管理器或 **virsh** 来启动、停止或暂停 VM Guest。您还可以将 VM Guest 配置为在引导 VM 主机服务器时自动启动。

关闭 VM Guest 时，可将其正常关机或强制关机。后一种操作的效果等同于拔下物理主机上的电源插头，建议仅在没有其他办法时才这样做。强制关机可能导致 VM Guest 上的文件系统损坏或数据丢失。




提示：正常关机

要能够执行正常关机，必须将 VM Guest 配置为支持 **ACPI**。如果 Guest 是使用虚拟机管理器创建的，则可在 VM Guest 中使用 ACPI。

根据 Guest 操作系统，能够使用 ACPI 可能还不足以执行正常关机。在生产环境中使用 Guest 之前，强烈建议先对其进行关机和重引导测试。例如，openSUSE 或 SUSE Linux Enterprise Desktop 可能需要获得 PolKit 授权才能关机和重引导。确保已在所有 VM Guest 上关闭此策略。

如果在安装 Windows XP/Windows Server 2003 Guest 期间启用了 ACPI，只在 VM Guest 配置中开启 ACPI 并不足够。有关更多信息，请参见：

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

无论 VM Guest 的配置如何，始终都可以从 Guest 操作系统内部实现正常关机。

10.3.1 使用虚拟机管理器更改 VM Guest 的状态

可以通过虚拟机管理器的主窗口或 VNC 窗口更改 VM Guest 的状态。

过程 10.1：通过虚拟机管理器窗口更改状态

1. 右键单击某个 VM Guest 项。
2. 在弹出菜单中选择运行、暂停或其中一个关机选项。

过程 10.2：通过 VNC 窗口更改状态

1. 按照第 10.2.1.1 节“使用虚拟机管理器打开图形控制台”中所述打开 VNC 窗口。
2. 在工具栏或虚拟机菜单中，选择运行、暂停或其中一个关机选项。

10.3.1.1 自动启动 VM Guest

您可以在引导 VM 主机服务器时自动启动 Guest。此功能默认未启用，需要为每个 VM Guest 单独启用。无法全局激活此功能。

1. 在虚拟机管理器中双击 VM Guest 项以打开其控制台。
2. 选择视图 > 细节打开 VM Guest 配置窗口。
3. 选择引导选项，然后选中在主机引导时启动虚拟机。
4. 单击应用保存新配置。

10.3.2 使用 **virsh** 更改 VM Guest 的状态

以下示例会更改名为“sles12”的 VM Guest 的状态。

开始

```
tux > virsh start sles12
```

暂停

```
tux > virsh suspend sles12
```

继续（已暂停的 VM Guest）

```
tux > virsh resume sles12
```

重引导

```
tux > virsh reboot sles12
```

正常关机

```
tux > virsh shutdown sles12
```

强制关机

```
tux > virsh destroy sles12
```

开启自动启动

```
tux > virsh autostart sles12
```

关闭自动启动

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

10.4 保存和恢复 VM Guest 的状态

保存 VM Guest 会保留其内存的确切状态。该操作类似于将计算机休眠。保存的 VM Guest 可以快速恢复到以前保存的运行中状况。

保存时，VM Guest 将会暂停，其当前内存状态将保存到磁盘，然后该 Guest 停止。该操作不会复制 VM Guest 虚拟磁盘的任何一部分。保存虚拟机所需的时长取决于分配的内存量。保存时，VM Guest 的内存将返回到 VM 主机服务器上的可用内存池。

恢复操作将装载先前保存的 VM Guest 内存状态文件并启动该 Guest。该 Guest 不会引导，而是在以前保存它的位置继续运行。该操作类似于退出休眠状态。

VM Guest 将保存到某个状态文件。请确保要保存到的分区有足够的空间。要估计预期的文件大小（以 MB 为单位），请在 Guest 上发出以下命令：

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



警告：始终恢复保存的 Guest

使用保存操作后，请不要引导或启动保存的 VM Guest。这样做会使计算机的虚拟磁盘和保存的内存状态不同步，有可能导致在恢复 Guest 时发生严重错误。

要能够再次使用保存的 VM Guest，请使用恢复操作。如果您是使用 **virsh** 保存 VM Guest 的，将无法使用虚拟机管理器将其恢复。在这种情况下，请务必使用 **virsh** 进行恢复。

❗ 重要：仅适用于磁盘类型为 raw、qcow2 的 VM Guest

仅当 VM Guest 使用的虚拟磁盘为 **raw** (**.img**) 或 **qcow2** 类型时，才可以保存和恢复该 VM Guest。

10.4.1 使用虚拟机管理器保存/恢复

过程 10.3：保存 VM GUEST

1. 打开 VM Guest 的 VNC 连接窗口。确保该 Guest 正在运行。
2. 选择虚拟机 > 关机 > 保存。

过程 10.4：恢复 VM GUEST

1. 打开 VM Guest 的 VNC 连接窗口。确保该 Guest 未运行。
2. 选择虚拟机 > 恢复。

如果 VM Guest 以前是使用虚拟机管理器保存的，则系统不会为您提供用于运行该 Guest 的选项。但请注意 **警告：始终恢复保存的 Guest** 中所述的有关使用 **virsh** 保存的计算机的注意事项。

10.4.2 使用 **virsh** 保存和恢复

使用 **virsh save** 命令保存运行中的 VM Guest，并指定要将其保存到的文件。

保存名为 **opensuse13** 的 Guest

```
tux > virsh save opensuse13 /virtual/saves/opensuse13.vmsav
```

保存 ID 为 **37** 的 Guest

```
tux > virsh save 37 /virtual/saves/opensuse13.vmsave
```

要恢复 VM Guest，请使用 `virsh restore`：

```
tux > virsh restore /virtual/saves/opensuse13.vmsave
```

10.5 创建和管理快照

VM Guest 快照是整个虚拟机的快照，包括 CPU、RAM、设备的状态，以及所有可写磁盘的内容。要使用虚拟机快照，所有挂接的硬盘均需使用 qcow2 磁盘映像格式，并且其中至少有一个硬盘需是可写的。

快照可让您将计算机恢复到特定时间点的状态。在撤消有错误的配置或者安装大量软件包时，此功能十分有用。启动一个在 VM Guest 处于关闭状态下创建的快照后，需要引导该 Guest。在该时间点之后写入磁盘的所有更改都将在启动快照时丢失。



注意

快照仅在 KVM VM 主机服务器上受支持。

10.5.1 术语

有多个特定术语用于描述快照的类型：

内部快照

保存到原始 VM Guest 的 qcow2 文件中的快照。该文件包含保存的快照状态，以及自截取快照以来发生的更改。内部快照的主要优势是它们全都储存在一个文件中，因此方便在多个计算机之间复制或移动。

外部快照

创建外部快照时，会保存原始 qcow2 文件并将其设为只读，同时会创建一个新的 qcow2 文件用于存放更改。原始文件有时称为“后备”文件或“基础”文件，包含所有更改的新文件称为“覆盖”文件或“派生”文件。备份 VM Guest 时，外部快照很有用。但外部快照不受虚拟机管理器的支持，且无法直接通过 `virsh` 删除。有关 QEMU 中外部快照的详细信息，请参见第 31.2.4 节“有效操作磁盘映像”。

实时快照

当原始 VM Guest 正在运行时创建的快照。内部实时快照支持保存设备以及内存和磁盘状态，而使用 virsh 的外部实时快照则支持保存内存状态和/或磁盘状态。

脱机快照

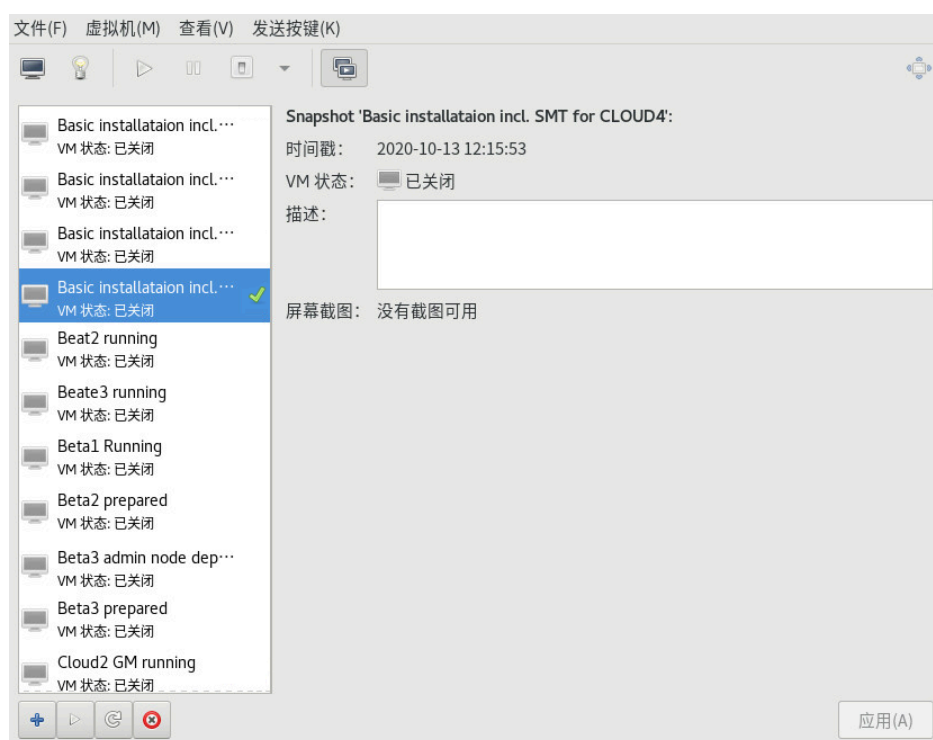
基于已关闭的 VM Guest 创建的快照。由于 Guest 的所有进程已停止且未使用任何内存，因此此类快照可确保数据完整性。

10.5.2 使用虚拟机管理器创建和管理快照

！ 重要：仅限内部快照

虚拟机管理器仅支持实时或脱机的内部快照。

要在虚拟机管理器中打开快照管理视图，请按照第 10.2.1.1 节“使用虚拟机管理器打开图形控制台”中所述打开 VNC 窗口。现在请选择视图 > 快照，或单击工具栏中的管理 VM 快照。



所选 VM Guest 的现有快照列表显示在窗口的左侧。上次启动的快照带有绿色对勾标记。窗口的右侧显示列表中当前标记的快照的细节。这些细节包括快照的标题和时戳、截取快照时 VM Guest 的状态以及说明。运行中 Guest 的快照还包括一个屏幕截图。可以直接在此视图中更改说明。其他快照数据不可更改。

10.5.2.1 创建快照

要截取 VM Guest 的新快照，请执行以下操作：

1. （可选）如果您要创建脱机快照，请关闭 VM Guest。
2. 单击 VNC 窗口左下角的添加。
创建快照窗口即会打开。
3. 提供名称和说明（可选）。截取快照后将无法更改该名称。为方便以后识别该快照，请使用“自述性名称”。
4. 单击完成以确认。

10.5.2.2 删除快照

要删除 VM Guest 的快照，请执行以下操作：

1. 单击 VNC 窗口左下角的删除。
2. 单击是确认删除。

10.5.2.3 启动快照

要启动快照，请执行以下操作：

1. 单击 VNC 窗口左下角的运行。
2. 单击是确认启动。

10.5.3 使用 **virsh** 创建和管理快照

要列出某个域（以下示例中为 `admin_server`）的所有现有快照，请运行 `snapshot-list` 命令：

```
tux > virsh snapshot-list --domain sle-ha-node1
Name                               Creation Time           State
-----
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
```

使用 `snapshot-current` 命令显示上次启动的快照：

```
tux > virsh snapshot-current --domain admin_server
Basic installation incl. SMT for CLOUD4
```

运行 `snapshot-info` 命令可以获取有关特定快照的细节：

```
tux > virsh snapshot-info --domain admin_server \
    -name "Basic installation incl. SMT for CLOUD4"
Name:          Basic installation incl. SMT for CLOUD4
Domain:        admin_server
Current:        yes
State:          shutoff
Location:       internal
Parent:         Basic installation incl. SMT for CLOUD3-HA
Children:       0
Descendants:     0
Metadata:       yes
```

10.5.3.1 创建内部快照

要截取 VM Guest 的内部快照（实时或脱机快照），请按如下所示使用 `snapshot-create-as` 命令：

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

- ❶ 域名。必需。
- ❷ 快照的名称。建议使用“自述性名称”，这样可以更轻松地区别该快照。必需。
- ❸ 快照的说明。可选。

10.5.3.2 创建外部快照

使用 `virsh` 可以截取 Guest 内存状态和/或磁盘状态的外部快照。

要同时截取 Guest 磁盘的实时和脱机外部快照，请指定 `--disk-only` 选项：

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

可以指定 `--diskspec` 选项来控制外部文件的创建方式：

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

要截取 Guest 内存的实时外部快照，请指定 `--live` 和 `--memspec` 选项：

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

要截取 Guest 磁盘和内存状态的实时外部快照，请将 `--live`、`--diskspec` 和 `--memspec` 选项结合使用：

```
tux > 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
```

有关更多细节，请参见 [man 1 virsh](#) 中的“快照命令”部分。

10.5.3.3 删除快照

无法使用 [virsh](#) 删除外部快照。要删除 VM Guest 的内部快照并恢复其占用的磁盘空间，请使用 [snapshot-delete](#) 命令：

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

10.5.3.4 启动快照

要启动快照，请使用 [snapshot-revert](#) 命令：

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

要启动当前快照（用于启动 VM Guest 的快照），使用 [--current](#) 便已足够，不需要指定快照名称：

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

10.6 删除 VM Guest

默认情况下，使用 [virsh](#) 删除 VM Guest 只会去除其 XML 配置。由于默认不会删除挂接的储存设备，因此您可以在另一个 VM Guest 上重复使用该储存设备。使用虚拟机管理器还可以删除 Guest 的储存文件 — 这会彻底擦除该 Guest。

10.6.1 使用虚拟机管理器删除 VM Guest

1. 在虚拟机管理器中，右键单击某个 VM Guest 项。
2. 从上下文菜单中选择删除。
3. 一个确认窗口即会打开。单击删除会永久擦除该 VM Guest。该删除操作不可恢复。

您还可以通过激活删除关联的储存文件来永久删除 Guest 的虚拟磁盘。该删除操作也不可恢复。

10.6.2 使用 **virsh** 删除 VM Guest

要删除 VM Guest，需先将其关闭。无法删除运行中的 Guest。有关关机的信息，请参见第 10.3 节 “更改 VM Guest 的状态：启动、停止、暂停”。

要使用 **virsh** 删除 VM Guest，请运行 **virsh undefine VM_NAME**。

```
tux > virsh undefine sles12
```

没有可自动删除挂接的储存文件的选项。如果这些文件由 libvirt 管理，请按照第 12.2.4 节 “从储存池中删除卷” 中所述将其删除。

10.7 迁移 VM Guest

虚拟化的主要优势之一是 VM Guest 可移植。当某台 VM 主机服务器出于维护目的而需要关闭时，或者当该主机过载时，可以轻松将 Guest 迁移到另一台 VM 主机服务器。KVM 和 Xen 甚至支持“实时”迁移，在此期间，VM Guest 仍然可用。

10.7.1 迁移要求

要成功将 VM Guest 迁移到另一台 VM 主机服务器，需符合以下要求：

- 建议源系统和目标系统采用相同的体系结构。
- 这两台计算机必须均可访问储存设备（例如，通过 NFS 或 iSCSI），并且必须将储存设备配置为这两台计算机上的储存池。有关更多信息，请参见第 12 章 “管理储存”。
在迁移期间连接的 CD-ROM 或软盘映像也需要符合此要求。不过，您可以按照第 14.11 节 “使用虚拟机管理器弹出和更换软盘或 CD/DVD-ROM 媒体” 中所述在迁移之前断开其连接。
- 这两台 VM 主机服务器上都需要运行 **libvirtd**，并且您必须能够打开从目标主机到源主机（或反向）的远程 **libvirt** 连接。有关详细信息，请参考第 11.3 节 “配置远程连接”。

- 如果目标主机上在运行防火墙，则需要打开端口以允许迁移。如果您在迁移期间不指定端口，`libvirt` 将从 49152:49215 范围内选择一个端口。确保在目标主机上的防火墙中打开此端口范围（建议）或所选的专用端口。
- 主机和目标计算机应位于网络上的同一子网中，否则迁移后网络功能无法正常工作。
- 参与迁移的所有 VM 主机服务器必须具有 `qemu` 用户的相同 UID，并具有 `kvm`、`qemu` 和 `libvirt` 组的相同 GID。
- 目标主机上不能存在正在运行或已暂停的同名 VM Guest。如果存在已关闭的同名计算机，其配置将被重写。
- 迁移 VM Guest 时支持除主机 CPU 型号以外的所有 CPU 型号。
- SATA 磁盘设备类型不可迁移。
- 文件系统直通功能不可迁移。
- 需在 VM 主机服务器和 VM Guest 上安装适当的计时功能。请参见第 18 章 “VM Guest 时钟设置”。
- 无法将物理设备从主机迁移到 Guest。目前，在使用具有 PCI 直通或 SR-IOV 功能的设备时，不支持实时迁移。如果需要使用软件虚拟化（半虚拟化或全虚拟化）。
- 缓存模式设置是重要的迁移设置。参见: 第 17.5 节 “缓存模式对实时迁移的影响”。
- 不支持向后迁移（例如，从 SLES 15 SP2 迁移到 15 SP1）。
- SUSE 正致力于支持将 VM Guest 从运行 LTSS 所涵盖的服务包的 VM 主机服务器，迁移到运行同一 SLES 主要版本中更新的服务包的 VM 主机服务器。例如，将 VM Guest 从 SLES 12 SP2 主机迁移到 SLES 12 SP5 主机。对于从 LTSS 迁移到更新的服务包的方案，SUSE 只会执行极简单的测试，建议在尝试迁移关键的 VM Guest 之前执行全面的现场测试。
- 在两台主机上，映像目录应位于同一路径中。
- 所有主机的微代码级别（尤其是 spectre 微代码更新）应该相同。在所有主机上安装 SUSE Linux Enterprise Server 的最新更新即可实现此目的。

10.7.2 使用虚拟机管理器进行迁移

使用虚拟机管理器迁移 VM Guest 时，在哪台计算机上启动虚拟机管理器并不重要。您可以在源主机或目标主机上启动虚拟机管理器，甚至可以在这两台之外的主机上启动。对于后一种情况，您需要能够同时与目标主机和源主机建立远程连接。

1. 启动虚拟机管理器，并与目标主机或源主机建立连接。如果虚拟机管理器不是在目标主机或源主机上启动的，则需要与这两台主机都建立连接。
2. 右键单击要迁移的 VM Guest，然后选择迁移。确保该 Guest 正在运行或已暂停 — 关闭的 Guest 无法迁移。



提示：提高迁移速度

要将迁移速度提高到一定程度，请暂停 VM Guest。这相当于使用前面所述的虚拟机管理器中的“脱机迁移”选项。

3. 为 VM Guest 选择一个新主机。如果所需的目标主机未显示，请确保您已连接到该主机。要更改用于连接到远程主机的默认选项，请在连接下设置模式，以及目标主机的地址（IP 地址或主机名）和端口。如果指定了端口，还必须指定地址。
在高级选项下，选择迁移是永久性的（默认设置）还是暂时性的（使用暂时移动）。此外我们还提供了一个选项允许不安全的迁移，它允许在不禁用 VM 主机服务器缓存的情况下进行迁移。这样可以加速迁移，但仅当当前配置能够在不使用 `cache="none" / 0_DIRECT` 的情况下提供一致的 VM Guest 储存信息时，此选项才起作用。



注意：带宽选项

在最近的虚拟机管理器版本中，去除了用于设置迁移带宽的选项。要设置特定的带宽，请改用 `virsh`。

4. 要执行迁移，请单击迁移。
迁移完成后，迁移窗口将会关闭，该 VM Guest 随即列在虚拟机管理器窗口中的新主机上。原始 VM Guest 仍可在目标主机上使用（处于关机状态）。

10.7.3 使用 **virsh** 进行迁移

要使用 **virsh migrate** 迁移 VM Guest，您需要能够直接访问或者通过外壳远程访问 VM 主机服务器，因为命令需在主机上运行。迁移命令如下所示：

```
tux > virsh migrate [OPTIONS] VM_ID_or_NAME CONNECTION_URI [--migrateuri
tcp://REMOTE_HOST:PORT]
```

下面列出了最重要的选项。有关完整列表，请参见 **virsh help migrate**。

--live

执行实时迁移。如果未指定此选项，Guest 将会在迁移期间暂停（“脱机迁移”）。

--suspend

执行脱机迁移，且不重启动目标主机上的 VM Guest。

--persistent

默认情况下，迁移的 VM Guest 只是暂时性迁移，因此 Guest 关闭后，系统会自动在目标主机上删除其配置。使用此开关可使迁移变为永久性迁移。

--undefinesource

如果指定此选项，成功迁移后，将删除源主机上的 VM Guest 定义（但不会删除挂接到此 Guest 的虚拟磁盘）。

--parallel --parallel-connections NUM_OF_CONNECTIONS

当单个迁移线程无法使源主机与目标主机之间的网络链接饱和时，可以使用并行迁移来提高迁移数据吞吐量。在具备 40 GB 网络接口的主机上，可能需要四个迁移线程才能使链接饱和。使用并行迁移可以大幅缩短迁移大内存 VM 所需的时间。

以下示例使用 mercury.example.com 作为源系统，使用 jupiter.example.com 作为目标系统；VM Guest 的名称为 opensuse131，ID 为 37。

使用默认参数进行脱机迁移

```
tux > virsh migrate 37 qemu+ssh://tux@jupiter.example.com/system
```

使用默认参数进行瞬态实时迁移

```
tux > virsh migrate --live opensuse131 qemu+ssh://tux@jupiter.example.com/
system
```

永久性实时迁移；删除源上的 VM 定义

```
tux > virsh migrate --live --persistent --undefinesource 37 \
qemu+tls://tux@jupiter.example.com/system
```

使用端口 49152 进行脱机迁移

```
tux > virsh migrate opensuse131 qemu+ssh://tux@jupiter.example.com/system \
--migrateuri tcp://@jupiter.example.com:49152
```



注意：瞬态迁移与永久性迁移的比较

默认情况下，**virsh migrate** 为目标主机上的 VM Guest 创建暂时性（瞬态）副本。

已关机版本的原始 Guest 说明将保留在源主机上。瞬态副本将会在 Guest 关机后从服务器中删除。

要为目标主机上的 Guest 创建永久副本，请使用开关 **--persistent**。已关机版本的原始 Guest 说明也会保留在源主机上。将选项 **--undefinesource** 与 **--persistent** 搭配使用可以实现“真正的”迁移，在此情况下，将在目标主机上创建永久副本，并删除源主机上的版本。

不建议只使用 **--undefinesource** 而不使用 **--persistent** 选项，因为这会导致两个 VM Guest 定义均会在目标主机上的 Guest 关闭后丢失。

10.7.4 分步操作示例

10.7.4.1 导出储存区

首先需要导出储存区，以便在主机之间共享 Guest 映像。可以通过一台 NFS 服务器完成此操作。在下面的示例中，我们想要共享网络 10.0.1.0/24 中所有计算机的 **/volume1/VM** 目录。我们将使用一台 SUSE Linux Enterprise NFS 服务器。以 root 用户身份编辑 **/etc/exports** 文件并添加以下内容：

```
/volume1/VM 10.0.1.0/24 (rw, sync, no_root_squash)
```

您需要重新启动该 NFS 服务器：

```
tux > sudo systemctl restart nfsserver
tux > sudo exportfs
/volume1/VM      10.0.1.0/24
```

10.7.4.2 在目标主机上定义池

在您要迁移 VM Guest 的每台主机上，必须定义池才能访问卷（其中包含 Guest 映像）。我们的 NFS 服务器 IP 地址为 10.0.1.99，它的共享是 /volume1/VM 目录，而我们想要将此共享装入 /var/lib/libvirt/images/VM 目录中。池名称为 VM。要定义此池，请创建包含以下内容的 VM.xml 文件：

```
<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>
```

然后使用 **pool-define** 命令将其装载到 libvirt：

```
root # virsh pool-define VM.xml
```

另一种定义此池的方法是使用 **virsh** 命令：

```
root # virsh pool-define-as VM --type netfs --source-host 10.0.1.99 \
      --source-path /volume1/VM --target /var/lib/libvirt/images/VM
Pool VM created
```

以下命令假设您在 **virsh** 的交互式外壳中操作，使用不带任何参数的 **virsh** 命令也可以访问该外壳。然后，可将池设置为在主机引导时自动启动（autostart 选项）：

```
virsh # pool-autostart VM
Pool VM marked as autostarted
```

如果您要禁用自动启动，请执行以下命令：

```
virsh # pool-autostart VM --disable
Pool VM unmarked as autostarted
```

检查该池是否存在：

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

virsh # pool-info VM
Name:          VM
UUID:          42efelb3-7eaa-4e24-a06a-ba7c9ee29741
State:         running
Persistent:    yes
Autostart:     yes
Capacity:      2,68 TiB
Allocation:    2,38 TiB
Available:     306,05 GiB
```



警告：池需要在所有目标主机上存在

请记住：必须在您要迁移其中的 VM Guest 的每台主机上定义此池。

10.7.4.3 创建卷

池已定义好，现在我们需要一个包含磁盘映像的卷。

```
virsh # vol-create-as VM sled12.qcow2 8G --format qcow2
```

```
Vol sled12.qcow2 created
```

稍后将会通过 `virt-install` 使用所示的卷名称安装 Guest。

10.7.4.4 创建 VM Guest

我们来使用 `virt-install` 命令创建一个 SUSE Linux Enterprise Server VM Guest。将使用 `--disk` 选项指定 VM 池；如果您不希望在执行迁移时使用 `--unsafe` 选项，建议您设置 `cache=none`。

```
root # virt-install --connect qemu:///system --virt-type kvm --name \
sled12 --memory 1024 --disk vol=VM/sled12.qcow2,cache=none --cdrom \
/mnt/install/ISO/SLE-12-Desktop-DVD-x86_64-Build0327-Media1.iso --graphics \
vnc --os-variant sled12
Starting install...
Creating domain...
```

10.7.4.5 迁移 VM Guest

一切准备就绪，现在可以执行迁移。在当前托管 VM Guest 的 VM 主机服务器上运行 `migrate` 命令，并选择目标。

```
virsh # migrate --live sled12 --verbose qemu+ssh://IP/Hostname/system
Password:
Migration: [ 12 %]
```

10.8 监视

10.8.1 使用虚拟机管理器进行监视

启动虚拟机管理器并连接到 VM 主机服务器后，所有运行中 Guest 的 CPU 使用率图表将会显示。

您也可以通过此工具获取有关磁盘和网络使用情况的信息，不过必须先**在首选项中激活此功能**：

1. 运行 `virt-manager`。
2. 选择编辑 > 首选项。
3. 从常规选项卡切换到轮询。
4. 激活您要查看的活动类型对应的复选框：轮询磁盘 I/O、轮询网络 I/O 和轮询内存统计信息。
5. 如果需要，还可以使用每隔 n 秒更新状态来更改更新间隔。
6. 关闭首选项对话框。
7. 在视图 > 图表下激活应显示的图表。

此后，磁盘和网络统计信息也会显示在虚拟机管理器的主窗口中。

可以从 VNC 窗口获取更精确的数据。按照第 10.2.1 节“[打开图形控制台](#)”中所述打开 VNC 窗口。在工具栏或视图菜单中选择细节。可以通过左侧树菜单中的性能项显示统计信息。

10.8.2 使用 `virt-top` 进行监视

`virt-top` 是一个命令行工具，与众所周知的进程监视工具 `top` 类似。`virt-top` 使用 `libvirt`，因此能够显示不同超级管理程序上运行的 VM Guest 的统计信息。建议使用 `virt-top`，而不要使用 `xentop` 等特定于超级管理程序的工具。

`virt-top` 默认会显示所有运行中 VM Guest 的统计信息。显示的数据包括已用内存百分比 (`%MEM`)、已用 CPU 百分比 (`%CPU`)，以及 Guest 的运行时长 (`TIME`)。数据会定期更新（默认为每三秒更新一次）。下面显示了某台 VM 主机服务器上的输出，该服务器包含七个 VM Guest，其中有四个处于非活动状态：

```
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 0:0 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)

```

输出默认按 ID 排序。使用以下组合键可以更改排序字段：

Shift + P：CPU 使用率

Shift + M：Guest 分配的内存总量

Shift + T：时间

Shift + I：ID

要使用任何其他字段进行排序，请按 **Shift + F** 并从列表中选择一个字段。要切换排序顺序，请使用 **Shift + R**。

virt-top 还支持基于 VM Guest 数据生成不同的视图，按以下键可以即时更改视图：

0：默认视图

1：显示物理 CPU

2：显示网络接口

3：显示虚拟磁盘

virt-top 支持使用更多热键来更改数据视图，并支持许多可以影响程序行为的命令行开关。

有关详细信息，请参见 [`man 1 virt-top`](#)。

10.8.3 使用 **kvm_stat** 进行监视

kvm_stat 可用于跟踪 KVM 性能事件。它会监视 `/sys/kernel/debug/kvm`，因此需要装入 debugfs。SUSE Linux Enterprise Server 上默认应该已装入 debugfs。如果未装入，请使用以下命令：

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

可采用三种不同的模式使用 **kvm_stat**：

```

kvm_stat          # update in 1 second intervals
kvm_stat -1       # 1 second snapshot
kvm_stat -l > kvmstats.log # update in 1 second intervals in log format
                        # can be imported to a spreadsheet

```

例 10.1：kvm_stat 的典型输出

```

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

有关如何解释这些值的更多信息，请参见 <http://clalance.blogspot.com/2009/01/kvm-performance-tools.html> 。

11 连接和授权

如果您要管理多个 VM 主机服务器，而每个服务器又托管了多个 VM Guest，那么管理工作很快就会变得困难起来。libvirt 的一个优势是，它能够一次连接到多个 VM 主机服务器，提供单个接口用于管理所有 VM Guest 以及连接其图形控制台。

为确保只有授权用户能够建立连接，libvirt 提供了多种可与不同授权机制（套接字、PolKit、SASL 和 Kerberos）结合使用的连接类型（通过 TLS、SSH、Unix 套接字和 TCP）。

11.1 身份验证

有权管理 VM Guest 和访问其图形控制台的用户应该限定在明确定义的人员范围内。为实现此目标，可在 VM 主机服务器上使用以下身份验证方法：

- 使用权限和组所有权对 Unix 套接字进行访问控制。此方法仅适用于 libvirtd 连接。
- 使用 PolKit 对 Unix 套接字进行访问控制。此方法仅适用于本地 libvirtd 连接。
- 使用 SASL（简单身份验证和安全层）进行用户名和口令身份验证。此方法适用于 libvirtd 和 VNC 连接。使用 SASL 不需要在服务器上拥有实际的用户帐户，因为 SASL 使用自己的数据库来储存用户名和口令。通过 SASL 进行身份验证的连接会加密。
- Kerberos 身份验证。此方法仅适用于 libvirtd 连接，本手册不予介绍。有关详细信息，请参考 http://libvirt.org/auth.html#ACL_server_kerberos。
- 单口令身份验证。此方法仅适用于 VNC 连接。

重要：libvirtd 和 VNC 的身份验证需要分开配置

对 VM Guest 管理功能的访问（通过 libvirtd）以及对其图形控制台的访问始终需要分开配置。限制对管理工具的访问时，这些限制不会自动应用到 VNC 连接。

通过 TLS/SSL 连接远程访问 VM Guest 时，可以通过仅允许特定的组拥有证书密钥文件的读取权限，在每个客户端上间接控制访问。有关详细信息，请参见第 11.3.2.5 节“限制访问（安全注意事项）”。

11.1.1 libvirtd 身份验证

`libvirtd` 身份验证在 `/etc/libvirt/libvirtd.conf` 中配置。此处进行的配置将应用到所有 `libvirt` 工具，例如虚拟机管理器或 `virsh`。

`libvirt` 提供了两个套接字：一个只读套接字用于监视目的，一个读写套接字用于管理操作。可以单独配置对这两个套接字的访问。默认情况下，这两个套接字由 `root.root` 拥有。默认仅向用户 `root` (`0700`) 授予对读写套接字的访问权限，而对于只读套接字，访问权限则完全开放 (`0777`)。

在以下说明中，您将了解如何配置对读写套接字的访问权限。这些说明同样也适用于只读套接字。所有配置步骤均需在 VM 主机服务器上执行。



注意：SUSE Linux Enterprise Server 上的默认身份验证设置

SUSE Linux Enterprise Server 上的默认身份验证方法是对 Unix 套接字进行访问控制。只有用户 `root` 能够进行身份验证。在 VM 主机服务器上以非 `root` 用户身份访问 `libvirt` 工具时，需要通过 `PolKit` 提供一次 `root` 口令。然后，系统将向您授予对当前和将来会话的访问权限。

或者，您可以配置 `libvirt`，以允许非特权用户进行“system”访问。有关详细信息，请参见第 11.2.1 节“非特权用户的“system”访问权限”。

建议的身份验证方法

本地连接

第 11.1.1.2 节 “使用 `PolKit` 对 Unix 套接字进行访问控制”

第 11.1.1.1 节 “使用权限和组所有权对 Unix 套接字进行访问控制”

基于 SSH 的远程隧道

第 11.1.1.1 节 “使用权限和组所有权对 Unix 套接字进行访问控制”

远程 TLS/SSL 连接

第 11.1.1.3 节 “使用 `SASL` 进行用户名和口令身份验证”

无（通过限制对证书的访问在客户端控制访问权限）

11.1.1.1 使用权限和组所有权对 Unix 套接字进行访问控制

要为非 root 帐户授予访问权限，请配置特定的组（在下面的示例中为 libvirt）拥有且可访问的套接字。此身份验证方法可用于本地和远程 SSH 连接。

1. 创建应该拥有套接字的组（如果不存在）：

```
tux > sudo groupadd libvirt
```

重要：组需要存在

在重新启动 libvirtd 之前，该组必须存在。否则，重新启动将会失败。

2. 将所需用户添加到该组：

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

3. 按如下所示在 /etc/libvirt/libvirtd.conf 中更改配置：

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

- ❶ 组所有权将设置给组 libvirt。
- ❷ 设置对套接字的访问权限 (srwxrwx---)。
- ❸ 禁用其他身份验证方法（PolKit 或 SASL）。访问仅通过套接字权限来控制。

4. 重新启动 libvirtd：

```
tux > sudo systemctl start libvirtd
```

11.1.1.2 使用 PolKit 对 Unix 套接字进行访问控制

对于 SUSE Linux Enterprise Server 上的非远程连接，使用 PolKit 对 Unix 套接字进行访问控制是默认的身份验证方法。因此，无需对 `libvirt` 配置进行更改。启用 PolKit 授权后，对上述两个套接字的权限将默认为 `0777`，每个尝试访问套接字的应用程序将需要通过 PolKit 进行身份验证。

重要：仅对本地连接进行 PolKit 身份验证

只能对 VM 主机服务器本身上的本地连接进行 PolKit 身份验证，因为 PolKit 不会处理远程身份验证。

有关访问 `libvirt` 套接字的策略有两个：

- `org.libvirt.unix.monitor`：访问只读套接字
- `org.libvirt.unix.manage`：访问读写套接字

默认情况下，有关访问读写套接字的策略是使用 `root` 口令进行一次身份验证，并授予对当前和将来的会话的特权。

要向用户授予无需提供 `root` 口令即可访问套接字的权限，您需要在 `/etc/polkit-1/rules.d` 中创建一条规则。创建包含以下内容的 `/etc/polkit-1/rules.d/10-grant-libvirt` 文件，以向组 `libvirt` 的所有成员授予对读写套接字的访问权限。

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

11.1.1.3 使用 SASL 进行用户名和口令身份验证

SASL 提供用户名和口令身份验证以及数据加密（默认为 `digest-md5`）。由于 SASL 会维护自己的用户数据库，VM 主机服务器上无需存在用户。TCP 连接需要 SASL，并且在 TLS/SSL 连接上也需要 SASL。

！ 重要：普通 TCP 以及提供 digest-md5 加密的 SASL

在未通过其他方式加密的 TCP 连接上使用 digest-md5 加密并不会为生产环境提供充足的安全性。建议仅在测试环境中使用这种加密。

💡 提示：在 TLS/SSL 上进行 SASL 身份验证

通过限制对证书密钥文件的访问，可以在客户端间接控制通过远程 TLS/SSL 连接进行的访问。但在处理大量客户端时，这种方法可能容易很出错。对 TLS 使用 SASL 可以另外在服务器端控制访问，从而提高安全性。

要配置 SASL 身份验证，请执行以下操作：

1. 按如下所示在 `/etc/libvirt/libvirtd.conf` 中更改配置：

a. 要为 TCP 连接启用 SASL，请使用以下配置：

```
auth_tcp = "sasl"
```

b. 要为 TLS/SSL 连接启用 SASL，请使用以下配置：

```
auth_tls = "sasl"
```

2. 重新启动 `libvirtd`：

```
tux > sudo systemctl restart libvirtd
```

3. libvirt SASL 配置文件位于 `/etc/sasl2/libvirtd.conf`。通常无需更改默认设置。

但是，如果在 TLS 上使用 SASL，您可以通过将设置 `mech_list` 参数的行注释掉，来关闭会话加密以避免额外的开销（TLS 连接已加密）。请仅对 TLS/SASL 执行此操作；对于 TCP 连接，此参数必须设置为 `digest-md5`。

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

4. 默认不会配置任何 SASL 用户，因此无法登录。使用以下命令可管理用户：

添加用户 tux

```
saslpaswd2 -a libvirt tux
```

删除用户 tux

```
saslpaswd2 -a libvirt -d tux
```

列出现有用户

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



提示：virsh 和 SASL 身份验证

使用 SASL 身份验证时，每次您发出 **virsh** 命令，都会收到输入用户名和口令的提示。在外壳模式下使用 **virsh** 可避免出现此提示。

11.1.2 VNC 身份验证

由于对 VM Guest 图形控制台的访问不是由 libvirt 控制，而是由特定的超级管理程序控制，因此始终需要另外配置 VNC 身份验证。主配置文件为 /etc/libvirt/<hypervisor>.conf。本节介绍的是 QEMU/KVM 超级管理程序，因此目标配置文件为 /etc/libvirt/qemu.conf。



注意：Xen 的 VNC 身份验证

相较于 KVM 和 LXC，Xen 只能按 VM 设置口令，无法提供更复杂的 VNC 身份验证。请参见下面的 <graphics type='vnc'... libvirt 配置选项。

可以使用两种身份验证类型：SASL 和单口令身份验证。如果您是使用 SASL 进行 libvirt 身份验证的，我们强烈建议也将它用于 VNC 身份验证 — 这样就可以共享同一数据库。

第三种限制对 VM Guest 的访问的方法是在 VNC 服务器上启用 TLS 加密。这要求 VNC 客户端有权访问 x509 客户端证书。通过限制对这些证书的访问，可以在客户端间接控制访问。有关详细信息，请参考第 11.3.2.4.2 节“基于 TLS/SSL 的 VNC：客户端配置”。

11.1.2.1 使用 SASL 进行用户名和口令身份验证

SASL 提供用户名和口令身份验证以及数据加密。由于 SASL 会维护自己的用户数据库，VM 主机服务器上无需存在用户。与对 `libvirt` 使用 SASL 身份验证一样，您可以在 TLS/SSL 连接上使用 SASL。有关配置这些连接的细节，请参见第 11.3.2.4.2 节“基于 TLS/SSL 的 VNC：客户端配置”。

要为 VNC 配置 SASL 身份验证，请执行以下操作：

1. 创建 SASL 配置文件。建议使用现有的 `libvirt` 文件。如果您已经为 `libvirt` 配置 SASL，并打算使用相同的设置（包括同一用户名和口令数据库），则使用简单的链接即可：

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

如果您只是为 VNC 设置 SASL，或者打算使用与 `libvirt` 不同的配置，请复制现有文件以用作模板：

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

然后根据需要编辑该文件。

2. 按如下所示在 `/etc/libvirt/qemu.conf` 中更改配置：

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

第一个参数使 VNC 监听所有公共接口（而不仅仅是本地主机），第二个参数启用 SASL 身份验证。

3. 默认不会配置任何 SASL 用户，因此无法登录。使用以下命令可管理用户：

添加用户 tux

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

删除用户 tux

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

列出现有用户

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

4. 重新启动 libvirtd:

```
tux > sudo systemctl restart libvirtd
```

5. 重新启动在更改配置之前已在运行的所有 VM Guest。未重新启动的 VM Guest 将不会对 VNC 连接使用 SASL 身份验证。



注意：支持的 VNC 查看器

目前，虚拟机管理器和 virt-viewer 均支持 SASL 身份验证。这两个查看器还支持 TLS/SSL 连接。

11.1.2.2 单口令身份验证

您也可以通过设置 VNC 口令来控制对 VNC 服务器的访问。可以为所有 VM Guest 设置一个全局口令，或者为每个 Guest 设置单独的口令。后一种做法需要编辑 VM Guest 的配置文件。



注意：始终设置全局口令

如果您要使用单口令身份验证，比较好的做法是设置一个全局口令，即使为每个 VM Guest 设置了口令也是如此。这样，在您忘记设置每个虚拟机的口令时，始终都可以通过一个“回退”口令保护您的虚拟机。仅当未为计算机设置其他口令时，才会使用全局口令。

过程 11.1：设置全局 VNC 口令

1. 按如下所示在 `/etc/libvirt/qemu.conf` 中更改配置：

```
vnc_listen = "0.0.0.0"
vnc_password = "PASSWORD"
```

第一个参数使 VNC 监听所有公共接口（而不仅仅是本地主机），第二个参数设置口令。口令的最大长度为八个字符。

2. 重启动 `libvirtd`：

```
tux > sudo systemctl restart libvirtd
```

3. 重启动在更改配置之前已在运行的所有 VM Guest。未重启动的 VM Guest 将不会对 VNC 连接使用口令身份验证。

过程 11.2：设置 VM GUEST 特定的 VNC 口令

1. 按如下所示在 `/etc/libvirt/qemu.conf` 中更改配置，以使 VNC 监听所有公共接口（而不仅仅是本地主机）。

```
vnc_listen = "0.0.0.0"
```

2. 在编辑器中打开 VM Guest 的 XML 配置文件。请将下面的示例中的 `VM_NAME` 替换为 VM Guest 的名称。使用的编辑器默认为 `$EDITOR`。如果未设置该变量，则使用 `vi`。

```
tux > virsh edit VM_NAME
```

3. 搜索具有 `type='vnc'` 属性的 `<graphics>` 元素，例如：

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

4. 添加 `passwd=PASSWORD` 属性，然后保存文件并退出编辑器。口令的最大长度为八个字符。

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

5. 重启动 `libvirtd`：

```
tux > sudo systemctl restart libvirtd
```

6. 重启动在更改配置之前已在运行的所有 VM Guest。未重启动的 VM Guest 将不会对 VNC 连接使用口令身份验证。



警告：VNC 协议的安全性

VNC 被认为是不安全的协议。尽管口令是以加密方式发送的，但如果攻击者可以嗅探到已加密的口令和加密密钥，该口令可能就会被利用。因此，建议将 VNC 与 TLS/SSL 结合使用，或通过 SSH 建立隧道。**virt-viewer**、虚拟机管理器和 Remmina（请参见《管理指南》，第 10 章“使用 VNC 的远程图形会话”，第 10.2 节“Remmina：远程桌面客户端”）支持这两种方法。

11.2 连接到 VM 主机服务器

要使用 **libvirt** 连接到超级管理程序，需要指定统一资源标识符 (URI)。使用 **virsh** 和 **virt-viewer** 时需要此 URI（在 VM 主机服务器上以 **root** 身份操作时例外），而使用虚拟机管理器时，此 URI 为可选项。虽然可以使用连接参数（例如 **virt-manager -c qemu:///system**）来调用虚拟机管理器，但虚拟机管理器还是提供了一个图形界面用于创建连接 URI。有关详细信息，请参见第 11.2.2 节“使用虚拟机管理器管理连接”。

HYPERVISOR ① + PROTOCOL ② : // USER@REMOTE ③ / CONNECTION_TYPE ④

- ① 指定超级管理程序。SUSE Linux Enterprise Server 目前支持以下超级管理程序：**test**（用于测试的虚设程序）、**qemu** (KVM) 和 **xen** (Xen)。此参数是必需的。
- ② 连接到远程主机时，请在此处指定协议。其值可以是：**ssh**（通过 SSH 隧道连接）、**tcp**（使用 SASL/Kerberos 身份验证进行 TCP 连接）或 **tls**（进行 TLS/SSL 加密的连接并通过 x509 证书完成身份验证）。
- ③ 连接到远程主机时，请指定用户名和远程主机名。如果未指定用户名，将使用调用了该命令的用户名 (**\$USER**)。有关详细信息，请参见下文。对于 TLS 连接，需要完全按照 x509 证书指定主机名。

- ④ 连接到 QEMU/KVM 超级管理程序时，接受两种连接类型：system（完全访问权限）或 session（受限访问权限）。由于 SUSE Linux Enterprise Server 不支持 session 访问权限，因此本文档将重点介绍 system 访问权限。

超级管理程序连接 URI 示例

test:///default

连接到本地虚设超级管理程序。用于测试。

qemu:///system 或 xen:///system

连接到本地主机上拥有完全访问权限（system 类型）的 QEMU/Xen 超级管理程序。

qemu+ssh://tux@mercury.example.com/system 或 xen+ssh://

tux@mercury.example.com/system

连接到远程主机 mercury.example.com 上的 QEMU/Xen 超级管理程序。连接是通过 SSH 隧道建立的。

qemu+tls://saturn.example.com/system 或 xen+tls://saturn.example.com/

system

连接到远程主机 mercury.example.com 上的 QEMU/Xen 超级管理程序。连接是使用 TLS/SSL 建立的。

有关更多细节和示例，请参见 <http://libvirt.org/uri.html> 上的 libvirt 文档。



注意：URI 中的用户名

使用 Unix 套接字身份验证时，需要指定用户名（无论是使用用户/口令身份验证模式还是 PolKit）。这适用于所有 SSH 连接和本地连接。

使用 SASL 身份验证（对于 TCP 或 TLS 连接）时或者不对 TLS 连接执行额外的服务器端身份验证时，无需指定用户名。使用 SASL 时不会评估用户名 — 在任何情况下，系统都会提示您输入 SASL 用户/口令组合。

11.2.1 非特权用户的“system”访问权限

如上文所述，可以使用两种不同的协议来与 QEMU 超级管理程序建立连接：session 和 system。“session”连接建立时具有与客户端程序相同的特权。此类连接适用于桌面虚拟化，因为它会受到限制（例如，无 USB/PCI 设备指派、无虚拟网络设置，只能对 libvirtd 进行受限的远程访问）。

适用于服务器虚拟化的“system”连接不存在功能限制，但默认仅可供 root 访问。不过，在将 DAC（自主访问控制）驱动程序添加到 libvirt 后，现在可以向非特权用户授予“system”访问权限。要向用户 tux 授予“system”访问权限，请执行以下操作：

过程 11.3：向普通用户授予“SYSTEM”访问权限

1. 按照第 11.1.1.1 节“使用权限和组所有权对 Unix 套接字进行访问控制”中所述通过 Unix 套接字启用访问权限。该示例中向 libvirt 组的所有成员都授予了 libvirt 访问权限，并使 tux 成为此组的成员。这样可确保 tux 能够使用 virsh 或虚拟机管理器进行连接。
2. 编辑 /etc/libvirt/qemu.conf 并按如下所示更改配置：

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

这样可确保 VM Guest 由 tux 启动，并且 tux 能够访问和修改已绑定至 Guest 的资源（例如虚拟磁盘）。

3. 使 tux 成为 kvm 组的成员：

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

需要执行此步骤才能授予对 /dev/kvm 的访问权限，而要启动 VM Guest 就必须具有此访问权限。

4. 重新启动 libvirtd：

```
tux > sudo systemctl restart libvirtd
```

11.2.2 使用虚拟机管理器管理连接

虚拟机管理器对它管理的每个 VM 主机服务器都使用一个 连接。每个连接都包含相应主机上的所有 VM Guest。默认已配置并已建立与本地主机的连接。

配置的所有连接都显示在虚拟机管理器主窗口中。活动连接带有小三角形标记，单击该标记可以收起或展开此连接的 VM Guest 列表。

非活动连接以灰色列出，带有 未连接 标记。可以双击或者右键单击这些连接，然后从上下文菜单中选择连接。还可以通过此菜单删除现有连接。



注意：编辑现有连接

无法编辑现有的连接。要更改连接，请使用所需参数创建新连接，然后删除“旧”连接。

要在虚拟机管理器中添加新连接，请执行以下操作：

1. 选择文件 > 添加连接
2. 选择主机的超级管理程序（Xen 或 QEMU/KVM）
3. （可选）要设置远程连接，请选择连接到远程主机。有关更多信息，请参见第 11.3 节“配置远程连接”。

如果要设置远程连接，请以 USERNAME@REMOTE _HOST 格式指定远程计算机的主机名。



重要：指定用户名

无需为 TCP 和 TLS 连接指定用户名：使用这些连接时，系统不会评估用户名。但在使用 SSH 连接时，如果您要以非 root 用户身份进行连接，则必须指定用户名。

4. 如果您不希望在启动虚拟机管理器时自动启动连接，请停用自动连接。
5. 单击连接以完成配置。

11.3 配置远程连接

`libvirt` 的一大优势是能够从一个中心位置管理不同远程主机上的 VM Guest。本节提供有关如何配置服务器和客户端以允许远程连接的详细说明。

11.3.1 基于 SSH 的远程隧道 (`qemu+ssh` 或 `xen+ssh`)

只需能够接受 SSH 连接，即可在 VM 主机服务器上启用基于 SSH 的远程隧道连接。确保 SSH 守护程序已启动 (`systemctl status sshd`)，并且已在防火墙中打开服务 SSH 的端口。

可以按照第 11.1.1.1 节 “使用权限和组所有权对 Unix 套接字进行访问控制” 中所述，使用传统的文件用户/组所有权和权限执行 SSH 连接的用户身份验证。以用户 `tux` 的身份进行连接 (`qemu+ssh://tuxsIVname;/system` 或 `xen+ssh://tuxsIVname;/system`) 是开箱即用功能，无需在 `libvirt` 一端进行额外的配置。

通过 SSH 进行连接 (`qemu+ssh://USER@SYSTEM` 或 `xen+ssh://USER@SYSTEM`) 时，需要提供 `USER` 的口令。按照《安全指南》，第 18 章 “SSH：安全性网络操作”，第 18.5.2 节 “复制 SSH 密钥” 中所述将公共密钥复制到 VM 主机服务器上的 `~USER/.ssh/authorized_keys` 可以避免此情况。在发出连接的计算机上使用 `ssh-agent` 会更方便。有关更多信息，请参见《安全指南》，第 18 章 “SSH：安全性网络操作”，第 18.5.3 节 “使用 `ssh-agent`”。

11.3.2 使用 x509 证书进行远程 TLS/SSL 连接 (`qemu+tls` 或 `xen+tls`)

与使用 SSH 相比，使用通过 x509 证书实现 TLS/SSL 加密和身份验证的 TCP 连接在设置上要复杂得多，不过此方法的可缩放性也高得多。如果您需要管理多个 VM 主机服务器，而这些服务器的管理员数量各异，请使用此方法。

11.3.2.1 基本概念

TLS（传输层安全）使用证书来加密两台计算机之间的通讯。发起连接的计算机一律视为“客户端”，使用的是“客户端证书”；接收方计算机一律视为“服务器”，使用的是“服务器证书”。例如，如果您通过一个中心桌面来管理 VM 主机服务器，则此方案适用。

如果连接是从两台计算机发起的，则每台计算机都需要有一个客户端证书和一个服务器证书。例如，如果您将 VM Guest 从一台主机迁移到另一台主机，就需要符合这种要求。

每个 x509 证书都有一个匹配的私用密钥文件。只有证书与私用密钥文件的组合才能正确标识自身。为确保证书由设想的拥有者颁发，该证书需由称为证书颁发机构 (CA) 的中心证书签名并颁发。客户端证书和服务器证书必须由同一个 CA 颁发。

！ 重要：用户身份验证

使用远程 TLS/SSL 连接只能确保允许两台计算机进行特定方向的通讯。通过限制对证书的访问，可以在客户端间接地限制只有特定用户可进行访问。有关更多信息，请参见第 11.3.2.5 节“限制访问（安全注意事项）”。

libvirt 还支持使用 SASL 在服务器上对用户进行身份验证。有关更多信息，请参见第 11.3.2.6 节“对 TLS 套接字使用 SASL 进行集中式用户身份验证”。

11.3.2.2 配置 VM 主机服务器

VM 主机服务器是接收连接的计算机，因此需要安装服务器证书，还需要安装 CA 证书。准备好证书后，可以为 libvirt 开启 TLS 支持。

1. 创建服务器证书，然后将其连同相应的 CA 证书一并导出。
2. 在 VM 主机服务器上创建以下目录：

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

按如下所示安装证书：

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


❗ 重要：限制对证书的访问

确保按照第 11.3.2.5 节“限制访问（安全注意事项）”中所述限制对证书的访问。

3. 通过编辑 `/etc/libvirt/libvirtd.conf` 并设置 `listen_tls = 1` 启用 TLS 支持。重启 `libvirtd`：

```
tux > sudo systemctl restart libvirtd
```

4. 默认情况下，`libvirt` 使用 TCP 端口 16514 来接受 TLS 安全连接。请在防火墙中打开此端口。

❗ 重要：在启用了 TLS 的情况下重启 `libvirtd`

如果您为 `libvirt` 启用了 TLS，则需要准备好服务器证书，否则重启 `libvirtd` 会失败。如果您更改了证书，也需要重启 `libvirtd`。

11.3.2.3 配置客户端并测试设置

客户端是发起连接的计算机，因此需要安装客户端证书，还需要安装 CA 证书。

1. 创建客户端证书，然后将其连同相应的 CA 证书一并导出。
2. 在客户端上创建以下目录：

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

按如下所示安装证书：

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

❗ 重要：限制对证书的访问

确保按照第 11.3.2.5 节“限制访问（安全注意事项）”中所述限制对证书的访问。

3. 发出以下命令测试客户端/服务器设置。请将 `mercury.example.com` 替换为您的 VM 主机服务器名称。指定在创建服务器证书时所用的完全限定主机名。

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

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

如果您的设置正确，您将会看到 VM 主机服务器上已在 `libvirt` 中注册的所有 VM Guest。

11.3.2.4 为 TLS/SSL 连接启用 VNC

目前只能通过几个工具来支持基于 TLS 的 VNC 通讯。`tightvnc` 或 `tigervnc` 等常见 VNC 查看器不支持 TLS/SSL。唯一可替代虚拟机管理器和 `virt-viewer` 的工具是 `remmina`（请参见《管理指南》，第 10 章“使用 VNC 的远程图形会话”，第 10.2 节“Remmina：远程桌面客户端”）。

11.3.2.4.1 基于 TLS/SSL 的 VNC：VM 主机服务器配置

要通过基于 TLS/SSL 的 VNC 访问图形控制台，需按如下所述配置 VM 主机服务器：

1. 在防火墙中打开服务 `VNC` 的端口。
2. 创建目录 `/etc/pki/libvirt-vnc`，并按如下所示将证书链接到此目录：

```
tux > sudo mkdir -p /etc/pki/libvirt-vnc && cd /etc/pki/libvirt-vnc
tux > sudo ln -s /etc/pki/CA/cacert.pem ca-cert.pem
tux > sudo ln -s /etc/pki/libvirt/servercert.pem server-cert.pem
tux > sudo ln -s /etc/pki/libvirt/private/serverkey.pem server-key.pem
```

3. 编辑 `/etc/libvirt/qemu.conf` 并设置以下参数：

```
vnc_listen = "0.0.0.0"
vnc_tls = 1
vnc_tls_x509_verify = 1
```

4. 重新启动 `libvirtd`:

```
tux > sudo systemctl restart libvirtd
```

! 重要：需要重新启动 VM Guest

只有在启动 VM Guest 时才会设置 VNC TLS。因此，需要重新启动更改配置前已在运行的所有计算机。

11.3.2.4.2 基于 TLS/SSL 的 VNC：客户端配置

需要在客户端执行的唯一操作是，将 x509 客户端证书放到可由所选客户端识别的位置。遗憾的是，虚拟机管理器和 **virt-viewer** 要求将证书放到不同的位置。虚拟机管理器可以从应用到所有用户的系统范围位置或者从每个用户的位置读取数据。在初始化与远程 VNC 会话的连接时，Remmina（请参见《管理指南》，第 10 章“使用 VNC 的远程图形会话”，第 10.2 节“Remmina：远程桌面客户端”）会要求提供证书位置。

虚拟机管理器 (**virt-manager**)

要连接到远程主机，虚拟机管理器需要使用第 11.3.2.3 节“配置客户端并测试设置”中所述的设置。要能够通过 VNC 进行连接，还需要将客户端证书放到以下位置：

系统范围的位置

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

每个用户的位置

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

virt-viewer

virt-viewer 仅接受位于系统范围位置的证书：

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

！ 重要：限制对证书的访问

确保按照第 11.3.2.5 节 “限制访问（安全注意事项）” 中所述限制对证书的访问。

11.3.2.5 限制访问（安全注意事项）

每个 x509 证书由两个部分构成：公共证书和私用密钥。只有使用了这两个部分，客户端才能通过身份验证。因此，对客户端证书及其私用密钥拥有读取访问权限的任何用户都可以访问您的 VM 主机服务器。另一方面，具有完整服务器证书的任意计算机都可以假装是 VM 主机服务器。这种情况也许不是您希望发生的，因此至少需要尽可能地限制对私用密钥文件的访问。要控制对密钥文件的访问，最简单的方法就是使用访问权限。

服务器证书

服务器证书需可由 QEMU 进程读取。在 SUSE Linux Enterprise Server QEMU 上，通过 `libvirt` 工具启动的进程由 `root` 拥有，因此只要 `root` 能够读取证书便已足够：

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

如果您在 `/etc/libvirt/qemu.conf` 中更改了 QEMU 进程的所有权，则也需要调整密钥文件的所有权。

系统范围的客户端证书

要控制对可在系统范围使用的密钥文件的访问，请限制只有特定的组具有读取访问权限，这样只有该组的成员可以读取密钥文件。下面的示例创建了一个 `libvirt` 组，并将 `clientkey.pem` 文件及其父目录的组所有权设置为 `libvirt`。之后，限制只有拥有者和组具有访问权限。最后，将用户 `tux` 添加到 `libvirt` 组，如此该用户便可以访问密钥文件。

```
CERTPATH="/etc/pki/libvirt/"  
# create group libvirt
```

```
groupadd libvirt
# change ownership to user root and group libvirt
chown root.libvirt $CERTPATH/private $CERTPATH/clientkey.pem
# restrict permissions
chmod 750 $CERTPATH/private
chmod 640 $CERTPATH/private/clientkey.pem
# add user tux to group libvirt
usermod --append --groups libvirt tux
```

每个用户的证书

需要将用于通过 VNC 访问 VM Guest 图形控制台的用户特定客户端证书放在用户主目录下的 `~/.pki` 中。与 SSH 不同，使用这些证书的 VNC 查看器不会检查私用密钥文件的访问权限（举例而言）。因此，用户需自行负责确保密钥文件不可由其他人读取。

11.3.2.5.1 限制从服务器端的访问

默认情况下，具有相应客户端证书的每个客户端都可以连接到接受 TLS 连接的 VM 主机服务器。因此，可以按照第 11.1.1.3 节“使用 SASL 进行用户名和口令身份验证”中所述通过 SASL 来实施额外的服务器端身份验证。

还可以通过 DN（判别名）白名单来限制访问，这样只有其证书与白名单中的某个 DN 匹配的客户端才能建立连接。

将允许的 DN 列表添加到 `/etc/libvirt/libvirtd.conf` 中的 `tls_allowed_dn_list`。此列表可以包含通配符。请不要指定空列表，因为这会导致拒绝所有连接。

```
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=*" ]
```

使用以下命令获取证书的判别名：

```
tux > certtool -i --infile /etc/pki/libvirt/clientcert.pem | grep "Subject:"
```

更改配置后重启 `libvirtd`：

```
tux > sudo systemctl restart libvirtd
```

11.3.2.6 对 TLS 套接字使用 SASL 进行集中式用户身份验证

通过 TLS 无法进行直接的用户身份验证 — 只能按照第 11.3.2.5 节“限制访问（安全注意事项）”中所述，通过证书读取权限在每个客户端上间接处理这种身份验证。但是，如果您需要采用基于服务器的集中式用户身份验证，`libvirt` 还允许在 TLS 上使用 SASL（简单身份验证和安全层）实现直接的用户身份验证。有关配置细节，请参见第 11.1.1.3 节“使用 SASL 进行用户名和口令身份验证”。

11.3.2.7 查错

11.3.2.7.1 虚拟机管理器/virsh 无法连接到服务器

按给定的顺序完成以下检查：

这是防火墙的问题吗（需要在服务器上打开 TCP 端口 16514）？

启动虚拟机管理器/`virsh` 的用户是否可以读取客户端证书（证书和密钥）？

是否在连接中指定了与服务器证书中相同的完全限定主机名？

是否在服务器上启用了 TLS (`listen_tls = 1`)？

是否在服务器上重启了 `libvirtd`？

11.3.2.7.2 VNC 连接失败

确保您可以使用虚拟机管理器连接到远程服务器。如果可以连接，请检查是否在启用了 TLS 支持的情况下启动了服务器上的虚拟机。下面的示例中的虚拟机名称为 `sles`。

```
tux > ps ax | grep qemu | grep "\-name sles" | awk -F" -vnc " '{ print FS $2 }'
```

如果输出不是以类似于下面的字符串开头，则表示虚拟机启动时未启用 TLS 支持，必须将虚拟机重新启动。

```
-vnc 0.0.0.0:0,tls,x509verify=/etc/pki/libvirt
```

12 管理储存

在 VM 主机服务器本身上管理 VM Guest 时，您可以访问 VM 主机服务器的整个文件系统，以挂接或创建虚拟硬盘，或将现有映像挂接到 VM Guest。但通过远程主机管理 VM Guest 时无法做到这些。出于此原因，libvirt 支持可从远程计算机访问的所谓“储存池”。



提示：CD/DVD ISO 映像

如果希望能够远程访问 VM 主机服务器上的 CD/DVD ISO 映像，需要将这些映像也放在储存池中。

libvirt 可识别两种不同类型的储存资源：卷和池。

储存卷

储存卷是可指派到 Guest 的储存设备 — 虚拟磁盘或 CD/DVD/软盘映像。从物理上而言（在 VM 主机服务器上），它可以是块设备（分区、逻辑卷等）或文件。

储存池

储存池是 VM 主机服务器上可用于储存卷的储存资源，类似于台式计算机的网络储存。从物理上而言，它可分为以下类型之一：

文件系统目录 (dir)

用于存放映像文件的目录。文件可以是支持的磁盘格式之一（raw 或 qcow2），也可以是 ISO 映像。

物理磁盘设备 (disk)

使用整个物理磁盘作为储存空间。系统会为添加到池的每个卷创建一个分区。

预格式化的块设备 (fs)

指定要使用的分区，该分区与文件系统目录池（用于存放映像文件的目录）的使用方式相同。唯一的区别在于，使用文件系统目录时，libvirt 会负责装入设备。

iSCSI 目标 (iscsi)

在 iSCSI 目标上设置池。您需要先登录卷一次，才能将卷用于 [libvirt](#)。使用 YaST iSCSI 发起端来检测和登录卷，有关细节，请参见《储存管理指南》。不支持在 iSCSI 池中创建卷；每个现有逻辑单元号 (LUN) 都代表一个卷。每个卷/LUN 还需要一个有效的（空）分区表或磁盘标签，这样您才能使用该卷。如果没有分区表或磁盘标签，请使用 **fdisk** 添加：

```
~ # fdisk -cu /dev/disk/by-path/ip-192.168.2.100:3260-iscsi-
iqn.2010-10.com.example:[...]-lun-2
Device contains neither a valid DOS partition table, nor Sun, SGI
or OSF disklabel
Building a new DOS disklabel with disk identifier 0xc15cdc4e.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

LVM 卷组 (logical)

使用 LVM 卷组作为池。您可以使用预定义的卷组，或者通过指定要使用的设备来创建组。储存卷会创建为卷上的分区。



警告：删除基于 LVM 的池

在储存管理器中删除基于 LVM 的池时，也会删除卷组。这会导致池中储存的所有数据丢失且不可恢复！

多路径设备 (mpath)

目前，多路径支持仅限于向 Guest 指派现有设备。不支持从 [libvirt](#) 内部创建卷或配置多路径。

网络导出的目录 (netfs)

指定要使用的网络目录，该网络目录与文件系统目录池（用于存放映像文件的目录）的使用方式相同。唯一的区别在于，使用文件系统目录时，[libvirt](#) 会负责装入目录。支持的协议为 NFS 和 GlusterFS。

SCSI 主机适配器 (scsi)

SCSI 主机适配器的使用方式与 iSCSI 目标基本相同。我们建议使用基于 [/dev/disk/by-*](#) 的设备名称而不是 [/dev/sdX](#)。后者可能会发生变化（例如，添加或拆除硬盘时）。不支持在 iSCSI 池中创建卷。每个现有 LUN（逻辑单元号）都代表一个卷。



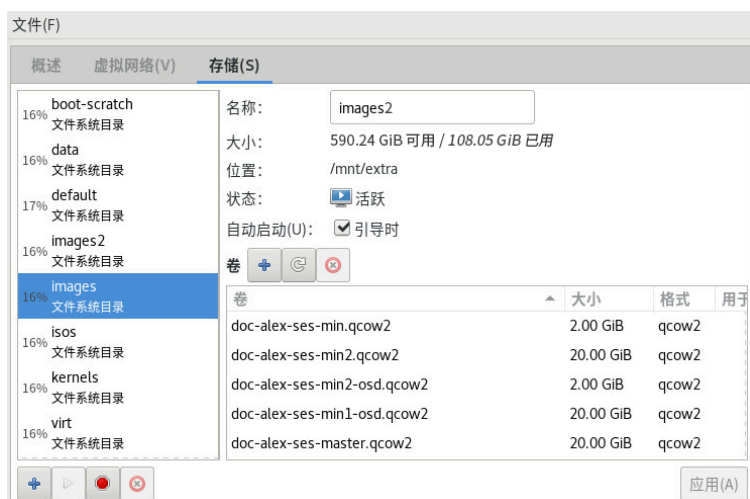
警告：安全考虑

为了避免数据丢失或损坏，请不要尝试使用也用于在 VM 主机服务器上构建储存池的资源，例如 LVM 卷组、iSCSI 目标等。无需从 VM 主机服务器连接到这些资源，也无需在 VM 主机服务器上装入这些资源 — [libvirt](#) 将负责这些事项。

不要在 VM 主机服务器上按标签装入分区。在某些情况下，分区可能是从 VM Guest 内部使用 VM 主机服务器上已有的名称标记的。

12.1 使用虚拟机管理器管理储存

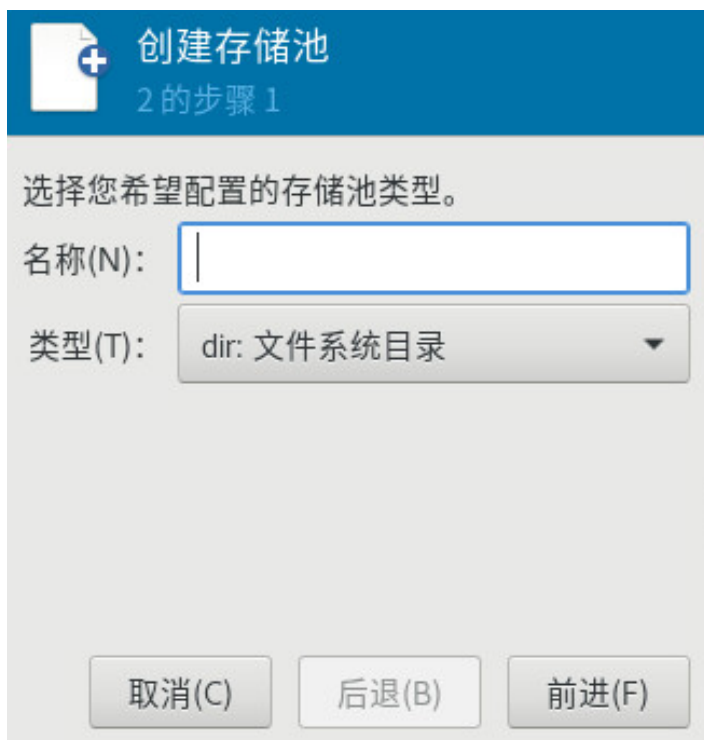
虚拟机管理器提供了一个图形界面，即储存管理器，用于管理储存卷和池。要访问储存管理器，请右键单击某个连接并选择细节，或者高亮显示某个连接并选择编辑 > 连接细节。选择储存选项卡。



12.1.1 添加储存池

要添加储存池，请执行以下操作：

1. 单击左下角的添加。添加新储存池对话框即会显示。
2. 提供池的名称（可包含字母数字字符和 `_`、`-`、`.`），然后选择类型。按继续可进行下一步。



3. 在随后出现的窗口中指定所需细节。需要输入的数据取决于您要创建的池类型：

类型：dir

- 目标路径：指定现有目录。

类型：disk

- 目标路径：用于存放设备的目录。一般情况下可以保留默认值 `/dev`。
- 格式：设备分区表的格式。一般情况下，使用 `auto` 应该不会有问题。如果有问题，请通过在 VM 主机服务器上运行命令 `parted -l` 获取所需的格式。
- 源路径：设备的路径。建议使用基于 `/dev/disk/by-*` 的设备名称而不是简单的 `/dev/sdX`，因为后者可能会发生变化（例如，添加或拆除硬盘时）。您需要指定类似整个磁盘而不是磁盘上的分区（如果存在）的路径。
- 构建池：激活此选项会格式化设备。设备上的所有数据都将丢失，请慎用！

类型：fs

- 目标路径：VM 主机服务器文件系统上的安装点。
- 格式：设备的文件系统格式。使用默认值 `auto` 应该不会有问题。
- 源路径：设备文件的路径。建议使用基于 `/dev/disk/by-*` 的设备名称而不是 `/dev/sdX`，因为后者可能会发生变化（例如，添加或拆除硬盘时）。

类型：iscsi

通过在 VM 主机服务器上运行以下命令来获取所需的数据：

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

此命令将返回采用以下格式的 iSCSI 卷列表。以粗体文本显示的元素是必需的：

```
IP_ADDRESS:PORT,TPGT TARGET_NAME_(IQN)
```

- 目标路径：包含设备文件的目录。使用 /dev/disk/by-path（默认值）或 /dev/disk/by-id。
- 主机名：iSCSI 服务器的主机名或 IP 地址。
- 源路径：iSCSI 目标名称 (IQN)。

类型：logical

- 目标路径：如果您使用现有卷组，请指定现有设备路径。构建新 LVM 卷组时，请指定 /dev 目录中目前不存在的设备名称。
- 源路径：使用现有卷组时，请留空。创建新卷组时，请在此处指定其设备。
- 构建池：仅当创建新卷组时才激活。

类型：mpath

- 目标路径：目前对多路径的支持仅限于通过它来使所有多路径设备可用。因此，请在此处指定任意字符串，此字符串随后将被忽略。必须指定路径，否则 XML 解析器将会失败。

类型：netfs

- 目标路径：VM 主机服务器文件系统上的安装点。
- 主机名：要导出网络文件系统的服务器的 IP 地址或主机名。
- 源路径：正在导出的服务器上的目录。

类型：scsi

- 目标路径：包含设备文件的目录。使用 /dev/disk/by-path（默认值）或 /dev/disk/by-id。
- 源路径：SCSI 适配器的名称。



注意：文件浏览

从远程位置操作时，无法通过单击浏览使用文件浏览器。

4. 单击完成以添加储存池。

12.1.2 管理储存池

您可以通过虚拟机管理器的储存管理器在池中创建或删除卷。您还可以暂时停用或永久删除现有的储存池。SUSE 目前不支持更改池的基本配置。

12.1.2.1 启动、停止和删除池

储存池的用途是提供 VM 主机服务器上的块设备，远程管理 VM Guest 时，可将这些设备添加到其中。要暂时禁止远程访问某个池，请单击储存管理器左下角的停止。停止的池标记为状态：非活动，并在列表窗格中灰显。默认情况下，新建的池在 VM 主机服务器引导时会自动启动。要启动某个非活动的池并再次使其可从远程位置使用，请单击储存管理器左下角的启动。



注意：池的状态不会影响挂接的卷

无论池处于什么状态（活动（已停止）或非活动（已启动）），池中挂接到 VM Guest 的卷都始终可用。池的状态只会影响到能否通过远程管理将卷挂接到 VM Guest。

要永久禁止访问某个池，请单击储存管理器左下角的删除。您只能删除非活动的池。删除某个池不会实际擦除其在 VM 主机服务器上的内容，而只会删除池配置。不过，在删除池时需要额外小心，尤其是删除基于 LVM 卷组的工具时：



警告：删除储存池

删除基于本地文件系统目录、本地分区或磁盘的储存池不会影响到这些池中当前挂接到 VM Guest 的卷的可用性。

如果删除 iSCSI、SCSI、LVM 组或网络导出的目录这些类型的池，将无法再从 VM Guest 访问这些池中的卷。尽管卷本身不会被删除，但 VM 主机服务器将不再可以访问这些资源。

创建足够大的新池或者直接从主机系统装入/访问这些资源时，iSCSI/SCSI 目标或网络导出的目录中的卷将再次可供访问。

删除基于 LVM 组的储存池时，将擦除 LVM 组定义，并且该 LVM 组将不再存在于主机系统上。其配置将不可恢复，并且此池中的所有卷都会丢失。

12.1.2.2 将卷添加到储存池

借助虚拟机管理器，您可以在所有储存池（多路径、iSCSI 或 SCSI 类型的池除外）中创建卷。这些池中的卷相当于 LUN，无法从 `libvirt` 内部更改。

1. 可以使用储存管理器创建新卷，或者在将新储存设备添加到 VM Guest 时创建新卷。在以上任一情况下，请从左侧面板中选择一个储存池，然后单击创建新卷。

2. 指定映像的名称并选择映像格式。

SUSE 目前仅支持 `raw` 或 `qcow2` 映像。后一个选项在基于 LVM 组的池中不可用。

在最大容量的旁边，指定允许磁盘映像达到的最大大小。除非您使用 `qcow2` 映像，否则还可以设置最初应该分配的分配量。如果这两个值不同，将会创建一个按需增长的稀疏映像文件。

对于 `qcow2` 映像，可以使用一个构成基本映像的后备储存（也称为“后备文件”）。这样新建的 `qcow2` 映像便只会记录对基本映像进行的更改。

3. 单击完成开始创建卷。

12.1.2.3 从储存池中删除卷

只能在储存管理器中删除卷：选择相应卷并单击删除卷。单击是确认。



警告：即使卷处于使用中状态，也能将其删除

即使活动或非活动的 VM Guest 中当前正在使用卷，也能将卷删除。无法恢复已删除的卷。

储存管理器中的使用对象列会指示某个卷是否已由 VM Guest 使用。

12.2 使用 **virsh** 管理储存

您也可以使用 **virsh** 通过命令行管理储存。不过，SUSE 目前不支持创建储存池。因此，本节仅限于介绍启动、停止和删除池以及卷管理等功能。

运行 **virsh help pool** 和 **virsh help volume** 可分别获取用于管理池和卷的所有 **virsh** 子命令列表。

12.2.1 列出池和卷

执行以下命令可以列出当前处于活动状态的所有池。要同时列出非活动池，请添加选项 **--all**：

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

可以使用 **pool-info** 子命令获取有关特定池的细节：

```
tux > virsh pool-info POOL
```

默认情况下，只能按池列出卷。要列出某个池中的所有卷，请输入以下命令。

```
tux > virsh vol-list --details POOL
```

目前，**virsh** 不提供任何可显示某个卷是否已由 Guest 使用的工具。下面的过程说明如何列出所有池中当前已被 VM Guest 使用的卷。

过程 12.1：列出当前已在 VM 主机服务器上使用的所有储存卷

1. 将以下内容保存到某个文件（例如 `~/libvirt/guest_storage_list.xml`）来创建一个 XSLT 样式表：

```
<?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. 在外壳中运行以下命令：假设 Guest 的 XML 定义全部储存在默认位置 (/etc/libvirt/qemu)。 **xsltproc** 由软件包 libxslt 提供。

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

12.2.2 启动、停止和删除池

使用 **virsh** pool 子命令来启动、停止或删除池。在以下示例中，请将 P00L 替换为池的名称或其 UUID：

停止池

```
tux > virsh pool-destroy P00L
```



注意：池的状态不会影响挂接的卷

无论池处于什么状态（活动（已停止）或非活动（已启动）），池中挂接到 VM Guest 的卷都始终可用。池的状态只会影响到能否通过远程管理将卷挂接到 VM Guest。

删除池

```
tux > virsh pool-delete P00L
```



警告：删除储存池

参见 [警告：删除储存池](#)

启动池

```
tux > virsh pool-start P00L
```

启用池自动启动功能

```
tux > virsh pool-autostart P00L
```

只有标记为 autostart 的池才会自动在 VM 主机服务器重引导时启动。

禁用池自动启动功能

```
tux > virsh pool-autostart P00L --disable
```

12.2.3 将卷添加到储存池

virsh 提供了两种将卷添加到储存池的方法：在 XML 定义中使用 `vol-create` 和 `vol-create-from` 添加，或者使用 `vol-create-as` 通过命令行参数添加。SUSE 目前不支持前一种方法，因此本节重点介绍 `vol-create-as` 子命令。

要将卷添加到现有池，请输入以下命令：

```
tux > virsh vol-create-as P00L ① NAME ② 12G --format ③ raw|qcow2 ④ --allocation 4G ⑤
```

- ① 卷要添加到的池的名称
- ② 卷的名称
- ③ 卷的大小，在本示例中为 12 GB。使用后缀 k、M、G、T 来分别表示千字节、兆字节、千兆字节和万亿字节。

- ④ 卷的格式。SUSE 目前支持 `raw` 和 `qcow2`。
- ⑤ 可选的参数。默认情况下，`virsh` 将创建一个按需增长的稀疏映像文件。使用此参数指定应分配的空间量（本示例中为 4 GB）。使用后缀 k、M、G、T 来分别表示千字节、兆字节、千兆字节和万亿字节。
如果不指定此参数，将生成不包含分配量的稀疏映像文件。要创建非稀疏卷，请使用此参数指定整个映像大小（在本示例中为 `12G`）。

12.2.3.1 克隆现有卷

将卷添加到池的另一种方法是克隆现有卷。请始终在原始实例所在的同一个池中创建新实例。

```
tux > virsh vol-clone NAME_EXISTING_VOLUME ① NAME_NEW_VOLUME ② --pool POOL ③
```

- ① 要克隆的现有卷的名称
- ② 新卷的名称
- ③ 可选的参数。`libvirt` 会尝试自动查找现有卷。如果找不到，请指定此参数。

12.2.4 从储存池中删除卷

要从池中永久删除某个卷，请使用 `vol-delete` 子命令：

```
tux > virsh vol-delete NAME --pool POOL
```

`--pool` 为可选项。`libvirt` 会尝试自动查找卷。如果找不到，请指定此参数。



警告：删除卷时不会进行检查

无论卷当前是否已在活动或非活动的 VM Guest 中使用，都将一律被删除。无法恢复已删除的卷。

只能使用[过程 12.1 “列出当前已在 VM 主机服务器上使用的所有储存卷”](#)中所述的方法来检测某个卷是否已由 VM Guest 使用。

12.2.5 将卷挂接到 VM Guest

按照第 12.2.3 节 “将卷添加到储存池” 中所述创建卷后，可将其挂接到虚拟机并作为硬盘使用：

```
tux > virsh attach-disk DOMAIN SOURCE_IMAGE_FILE TARGET_DISK_DEVICE
```

例如：

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

要检查是否已挂接新磁盘，请检查 `virsh dumpxml` 命令的结果：

```
root # 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>
[...]
```

12.2.5.1 热插入或持久更改

可将磁盘挂接到活动和非活动的域。挂接操作由 `--live` 和 `--config` 选项控制：

`--live`

将磁盘热插入到活动域。挂接操作不会保存在域配置中。对非活动域使用 `--live` 会出错。

`--config`

持久更改域配置。下一次启动域后，挂接的磁盘将可用。

`--live --config`

热插入磁盘并将其添加到持久域配置中。



提示：virsh attach-device

`virsh attach-device` 是 `virsh attach-disk` 的更通用的形式。可以使用此命令将其他类型的设备挂接到域。

12.2.6 从 VM Guest 分离卷

要从域分离磁盘，请使用 `virsh detach-disk`：

```
root # virsh detach-disk DOMAIN TARGET_DISK_DEVICE
```

例如：

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

可以按照第 12.2.5 节 “将卷挂接到 VM Guest” 中所述，使用 `--live` 和 `--config` 选项来控制挂接操作。

12.3 使用 virtlockd 锁定磁盘文件和块设备

锁定块设备和磁盘文件可以防止从不同的 VM Guest 并发向这些资源写入数据。它可以防范启动同一个 VM Guest 两次，或者将同一个磁盘添加到两个不同的虚拟机。这样就会减少由于配置错误导致虚拟机磁盘映像损坏的风险。

锁定操作由名为 `virtlockd` 的守护程序控制。由于此守护程序独立于 `libvirtd` 守护程序运行，在 `libvirtd` 崩溃或重新启动后，锁将会保留。甚至在 `virtlockd` 本身更新之后，锁也仍会保留，因为此守护程序能够自行重新执行。这可以确保当 `virtlockd` 更新后，无需重新启动 VM Guest。KVM、QEMU 和 Xen 支持 `virtlockd`。

12.3.1 启用锁定

SUSE Linux Enterprise Server 上默认未启用锁定虚拟磁盘的功能。要启用锁定并在系统重引导时自动启动锁定，请执行以下步骤：

1. 编辑 `/etc/libvirt/qemu.conf` 并设置

```
lock_manager = "lockd"
```

2. 使用以下命令启动 `virtlockd` 守护程序：

```
tux > sudo systemctl start virtlockd
```

3. 使用以下命令重新启动 `libvirtd` 守护程序：

```
tux > sudo systemctl restart libvirtd
```

4. 确保引导系统时自动启动 `virtlockd`：

```
tux > sudo systemctl enable virtlockd
```

12.3.2 配置锁定

`virtlockd` 默认配置为自动锁定为 VM Guest 配置的所有磁盘。默认设置使用“直接”锁空间，在这种情况下，系统会根据与 VM Guest <disk> 设备关联的实际文件路径获取锁。例如，如果 VM Guest 包含下面的 <disk> 设备，将直接针对 `/var/lib/libvirt/images/my-server/disk0.raw` 调用 `flock(2)`：

```
<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>
```

可以通过编辑 `/etc/libvirt/qemu-lockd.conf` 文件来更改 `virtlockd` 配置。此文件还包含详细注释及其他信息。确保通过重新装载 `virtlockd` 来激活配置更改：

```
tux > sudo systemctl reload virtlockd
```

12.3.2.1 启用间接锁空间

`virtlockd` 的默认配置使用“直接”锁空间。这意味着，系统会根据与 <disk> 设备关联的实际文件路径来获取锁。

如果磁盘文件路径不可供所有主机访问，可将 `virtlockd` 配置为允许“间接”锁空间。这意味着，系统会使用磁盘文件路径的哈希在间接锁空间目录中创建一个文件。然后，将在这些哈希文件而不是实际的磁盘文件路径中存放锁。如果包含磁盘文件的文件系统不支持 `fcntl()` 锁，也可以使用间接锁空间。使用 `file_lockspace_dir` 设置指定间接锁空间：

```
file_lockspace_dir = "/MY_LOCKSPACE_DIRECTORY"
```

12.3.2.2 在 LVM 或 iSCSI 卷上启用锁定

如果您要锁定由多个主机共享的 LVM 或 iSCSI 卷上的虚拟磁盘，则需要按 UUID 而不是路径（默认使用路径）执行锁定。此外，需将锁空间目录放在可供共享该卷的所有主机访问的共享文件系统上。为 LVM 和/或 iSCSI 设置以下选项：

```
lvm_lockspace_dir = "/MY_LOCKSPACE_DIRECTORY"
iscsi_lockspace_dir = "/MY_LOCKSPACE_DIRECTORY"
```

12.4 联机调整 Guest 块设备的大小

有时，您需要更改（扩展或收缩）Guest 系统使用的块设备的大小。例如，当最初分配的磁盘空间不再足够时，便需要增大空间大小。如果 Guest 磁盘驻留在逻辑卷中，您可以在 Guest 系统正在运行时调整该磁盘的大小。与脱机调整磁盘大小相比（请参见第 19.3 节“Guestfs 工具”软件包中的 `virt-resize` 命令），这是一项巨大的优势，因为 Guest 提供的服务在调整大小期间不会受到干扰。要调整 VM Guest 磁盘的大小，请执行以下步骤：

过程 12.2：联机调整 GUEST 磁盘的大小

1. 在 Guest 系统内部，检查磁盘（例如 `/dev/vda`）的当前大小。

```
root # 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. 在主机上，将容纳 Guest 磁盘 `/dev/vda` 的逻辑卷调整到所需大小，例如 200 GB。

```
root # lvresize -L 200G /dev/mapper/vg00-home
Extending logical volume home to 200 GiB
Logical volume home successfully resized
```

3. 在主机上，调整与 Guest 磁盘 `/dev/mapper/vg00-home` 相关的块设备的大小。请注意，您可以使用 `virsh list` 来查找 `DOMAIN_ID`。

```
root # virsh blockresize --path /dev/vg00/home --size 200G DOMAIN_ID
Block device '/dev/vg00/home' is resized
```

4. 检查 Guest 是否接受新磁盘大小。

```
root # 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
```

12.5 在主机与 Guest 之间共享目录（文件系统直通）

libvirt 允许使用 QEMU 的文件系统直通（也称为 VirtFS）功能在主机与 Guest 之间共享目录。此类目录还可由多个 VM Guest 同时访问，因此可用于在 VM Guest 之间交换文件。



注意：Windows Guest 和文件系统直通

请注意，无法通过文件系统直通在 VM 主机服务器与 Windows Guest 之间共享目录，因为 Windows 缺少装入共享目录所需的驱动程序。

要使共享目录可在 VM Guest 上使用，请执行以下操作：

1. 在虚拟机管理器中打开 Guest 的控制台，然后从菜单中选择视图 > 细节，或者在工具栏中单击显示虚拟硬件细节。选择添加硬件 > 文件系统打开文件系统直通对话框。

2. 可以在驱动程序中选择句柄或路径基本驱动程序。默认设置为路径。可以在模式中选择安全模型，这会影响在主机上设置文件权限的方式。有三个选项可用：

直通（默认设置）

使用客户端用户的身份凭证直接在文件系统上创建文件。这与 NFSv3 使用的设置非常相似。

Squash

与直通相同，但会忽略 `chown` 等特权操作的失败事件。当以 `root` 特权之外的身份运行 KVM 时，需要选择此选项。

已映射

使用文件服务器的身份凭证 (`qemu.qemu`) 创建文件。用户身份凭证和客户端用户的身份凭证保存在扩展属性中。当主机和 Guest 域应彻底隔离时，建议使用此模型。

3. 使用源路径指定 VM 主机服务器上的目录的路径。在目标路径中输入一个字符串，作为装入共享目录时使用的标记。请注意，此字段中的字符串仅作为标记，不是 VM Guest 上的路径。
4. 应用设置。如果 VM Guest 当前正在运行，需要将其关闭才能应用新设置（重引导 Guest 是不够的）。
5. 引导 VM Guest。要装入共享目录，请输入以下命令：


```
tux > sudo mount -t 9p -o trans=virtio,version=9p2000.L,rw TAG /MOUNT_POINT
```

要使共享目录永久可用，请将下面一行添加到 `/etc/fstab` 文件中：

```
TAG /MOUNT_POINT 9p trans=virtio,version=9p2000.L,rw 0 0
```

12.6 通过 libvirt 使用 RADOS 块设备

RADOS 块设备 (RBD) 将数据储存在 Ceph 群集中。这些设备支持快照、复制和数据一致性。您可以像使用其他块设备一样，从 `libvirt` 管理的 VM Guest 使用 RBD。

有关更多细节，请参见 SUSE Enterprise Storage 《Administration Guide》（管理指南）中的“Using libvirt with Ceph”（将 libvirt 与 Ceph 配合使用）一章。<https://documentation.suse.com/ses/>  中提供了 SUSE Enterprise Storage 文档。

13 管理网络

本章介绍 VM 主机服务器的常用网络配置，包括 VM 主机服务器和 `libvirt` 本机支持的配置。这些配置对 SUSE Linux Enterprise Server 支持的所有超级管理程序（例如 KVM 或 Xen）均有效。

有两个常用网络配置可为 VM Guest 提供网络连接：

- 充当第 2 层交换机的网桥
- 由 `libvirt` 管理且支持第 3 层转发的虚拟网络

13.1 网桥

网桥配置为 VM Guest 提供第 2 层交换机，可以基于与端口关联的 MAC 地址在网桥上的端口之间交换第 2 层以太网包。这样 VM Guest 便可通过第 2 层连接访问 VM 主机服务器的网络。此配置类似于将 VM Guest 的虚拟以太网网线连接到与主机以及主机上运行的其他 VM Guest 共享的集线器。该配置通常称为共享物理设备。

当 SUSE Linux Enterprise Server 配置为 KVM 或 Xen 超级管理程序时，网桥配置是它的默认配置。如果您只想将 VM Guest 连接到 VM 主机服务器的 LAN，则此为首选配置。

13.1.1 使用 YaST 管理网桥

本节包含使用 YaST 添加或删除网桥的过程。

13.1.1.1 添加网桥

要在 VM 主机服务器上添加网桥，请执行以下步骤：

1. 启动 YaST > 系统 > 网络设置。
2. 激活概述选项卡并单击添加。
3. 从设备类型列表中选择网桥，然后在配置名称项中输入网桥设备接口名称。按下一步按钮继续。

4. 在地址选项卡中指定网络细节，例如 DHCP/静态 IP 地址、子网掩码或主机名。
仅当您也将设备指派到与 DHCP 服务器连接的网桥时，动态地址才有作用。
如果您打算创建不与实际以太网设备连接的虚拟网桥，请使用静态指派的 IP 地址。在这种情况下，比较好的做法是使用私有 IP 地址范围（例如 192.168.0.0/16、172.16.0.0/12 或 10.0.0.0/8）内的地址。
要创建仅在不同 Guest 之间充当连接点，而不连接到主机系统的网桥，请将 IP 地址设置为 0.0.0.0，将子网掩码设置为 255.255.255.255。网络脚本会将此特殊地址作为未设置的 IP 地址来处理。
5. 激活桥接的设备选项卡，然后激活您要包含在网桥中的网络设备。
6. 单击下一步返回到概述选项卡，然后单击确定以确认。现在，新网桥应在 VM 主机服务器上处于活动状态。

13.1.1.2 删除网桥

要删除现有网桥，请执行以下步骤：

1. 启动 YaST > 系统 > 网络设置。
2. 从概述选项卡上的列表中选择您要删除的网桥设备。
3. 单击删除以删除该网桥，然后单击确定以确认。

13.1.2 通过命令行管理网桥

本节包含使用命令行添加或去除网桥的过程。

13.1.2.1 添加网桥

要在 VM 主机服务器上添加新网桥设备，请执行以下步骤：

1. 在要创建新网桥的 VM 主机服务器上以 root 身份登录。
2. 为新网桥选择一个名称（在本示例中为 virbr_test），然后运行以下命令

```
root # ip link add name VIRBR_TEST type bridge
```

3. 检查是否已在 VM 主机服务器上创建了网桥：

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

virbr_test 存在，但不与任何物理网络接口相关联。

4. 启动该网桥，并在其中添加一个网络接口：

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



重要：网络接口必须未被使用

只能指派尚未被其他网桥使用的网络接口。

5. （可选）启用 STP（请参见[生成树协议 \(https://en.wikipedia.org/wiki/Spanning_Tree_Protocol\)](https://en.wikipedia.org/wiki/Spanning_Tree_Protocol) ↗）

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

13.1.2.2 删除网桥

要通过命令行删除 VM 主机服务器上的现有网桥设备，请执行以下步骤：

1. 在要从中删除现有网桥的 VM 主机服务器上以 root 身份登录。
2. 列出现有网桥，以识别要去除的网桥的名称：

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

3. 删除网桥：

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

13.1.3 使用 VLAN 接口

有时需要在两台 VM 主机服务器之间或者 VM Guest 系统之间创建私用连接。例如，要将 VM Guest 迁移到不同网段中的主机，或者创建只有 VM Guest 系统能够连接到的私用网桥（即使这些 VM Guest 在不同的 VM 主机服务器系统上运行）。构建此类连接的简单方法是设置 VLAN 网络。

VLAN 接口通常在 VM 主机服务器上设置。它们可以将不同的 VM 主机服务器系统互连，或者可以设置为其他仅限虚拟连接的网桥的物理接口。甚至可以创建一个使用 VLAN 作为物理接口（该接口在 VM 主机服务器中没有 IP 地址）的网桥。这样，Guest 系统就无法通过此网络访问主机。

运行 YaST 模块系统 > 网络设置。执行以下过程设置 VLAN 设备：

过程 13.1：使用 YAST 设置 VLAN 接口

1. 单击添加以创建新网络接口。
2. 在硬件对话框中，选择设备类型 VLAN。
3. 将配置名称的值更改为 VLAN 的 ID。请注意，VLAN ID 1 通常用于管理目的。
4. 单击“下一步”。
5. 将 VLAN 设备应连接到的借口选择为下面的 VLAN 的实际接口。如果所需的接口未显示在列表中，请先在不指定 IP 地址的情况下设置此接口。
6. 选择将 IP 地址指派到 VLAN 设备的所需方法。
7. 单击下一步完成配置。

还可将 VLAN 接口用作网桥的物理接口。这样便可以连接多个仅限 VM 主机服务器的网络，以及实时迁移与此类网络连接的 VM Guest 系统。

YaST 并非始终允许不设置 IP 地址。但有时可能需要这种功能，尤其是应该连接仅限 VM 主机服务器的网络时。在这种情况下，请使用特殊地址 0.0.0.0 和网络掩码 255.255.255.255。系统脚本会将此地址当作未设置 IP 地址来处理。

13.2 虚拟网络

`libvirt` 管理的虚拟网络类似于桥接的网络，但通常不与 VM 主机服务器建立第 2 层连接。与 VM 主机服务器物理网络的连接通过第 3 层转发来实现，与第 2 层桥接网络相比，第 3 层转发在 VM 主机服务器上引入了额外的包处理。虚拟网络还为 VM Guest 提供 DHCP 和 DNS 服务。有关 `libvirt` 虚拟网络的详细信息，请参见 <https://libvirt.org/formatnetwork.html> 上的《Network XML format》（网络 XML 格式）文档。

SUSE Linux Enterprise Server 上的标准 `libvirt` 安装已经随附了名为 `default` 的预定义虚拟网络，该虚拟网络为网络提供 DHCP 和 DNS 服务，并且能够通过网络地址转换 (NAT) 转发模式连接到 VM 主机服务器的物理网络。尽管已预定义了 `default` 虚拟网络，但它必须由管理员显式启用。有关 `libvirt` 支持的转发模式的详细信息，请参见 <https://libvirt.org/formatnetwork.html#elementsConnect> 上《Network XML format》（网络 XML 格式）文档中的“Connectivity”（连接）一节。

`libvirt` 管理的虚拟网络可用于满足各种用例，但通常是在进行无线连接或动态/零星网络连接的 VM 主机服务器（例如便携式计算机）上使用。虚拟网络也适用于 VM 主机服务器网络的 IP 地址有限的情形，可用在虚拟网络与 VM 主机服务器的网络之间转发包。不过，大多数服务器用例更适合网桥配置，其中的 VM Guest 会连接到 VM 主机服务器的 LAN。



警告：启用转发模式

在 `libvirt` 虚拟网络中启用转发模式会通过将 `/proc/sys/net/ipv4/ip_forward` 和 `/proc/sys/net/ipv6/conf/all/forwarding` 设置为 1 在 VM 主机服务器中启用转发，而这本质上是将 VM 主机服务器转变成路由器。重新启动 VM 主机服务器的网络可能会重置值并禁用转发。要避免这种行为，请通过编辑 `/etc/sysctl.conf` 文件并添加以下设置，在 VM 主机服务器中显式启用转发：

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

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

13.2.1 使用虚拟机管理器管理虚拟网络

您可以使用虚拟机管理器定义、配置和操作虚拟网络。

13.2.1.1 定义虚拟网络

1. 启动虚拟机管理器。在可用连接列表中，右键单击您需要为其配置虚拟网络的连接名称，然后选择细节。
2. 在连接细节窗口中，单击虚拟网络选项卡。您会看到可用于当前连接的所有虚拟网络的列表。右侧会显示选定虚拟网络的细节。



图 13.1：连接细节

3. 要添加新虚拟网络，请单击添加。
4. 指定新虚拟网络的名称，然后单击前进。



创建虚拟网络
4 的步骤 1

为虚拟网络选择名称：

网络名称(N): vnet1

示例： network1

取消(C) 后退(B) 前进(F)

图 13.2：创建虚拟网络

5. 要指定 IPv4 网络地址空间定义，请激活相关选项，并在网络文本项中键入定义。



创建虚拟网络
4 的步骤 2

为虚拟网络选择 IPv4 地址空间：

☒ 启用 IPv4 网络地址空间定义

网络(N): 192.168.100.0/24

提示：网络地址应在 IPv4 私有地址范围内选择。例如：10.0.0.0/8 或 192.168.0.0/16

网关： 192.168.100.1

类型： 专用

☒ 启用 DHCPv4

开始： 192.168.100.128

结束： 192.168.100.254

☒ 启用静态路由

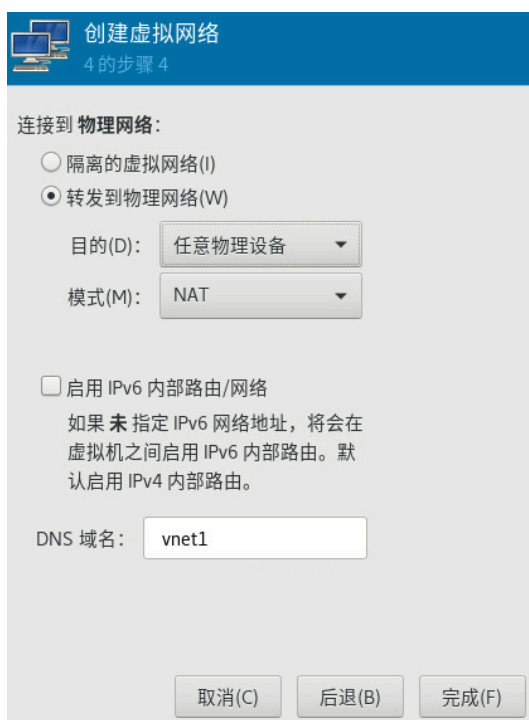
到网络： 10.10.20.0/24

通过 网关： 192.168.100.1

取消(C) 后退(B) 前进(F)

图 13.3：创建虚拟网络

6. `libvirt` 可为虚拟网络提供 DHCP 服务器。如果需要此服务器，请激活启用 DHCPv4，然后键入可指派地址的开始和结束 IP 地址范围。
7. 要为新虚拟网络启用静态路由，请激活相关选项，并键入网络地址和网关地址。
8. 单击前进继续。
9. 要指定 IPv6 相关的选项（网络地址空间、DHCPv6 服务器或静态路由），请激活启用 IPv6 网络地址空间定义，然后激活相关选项并填写相关的框。
10. 单击前进继续。
11. 选择您是要创建隔离的虚拟网络还是转发的虚拟网络。



创建虚拟网络
4 的步骤 4

连接到 物理网络：

☐ 隔离的虚拟网络(I)
☒ 转发到物理网络(W)

目的(D): 任意物理设备

模式(M): NAT

☐ 启用 IPv6 内部路由/网络
如果 未 指定 IPv6 网络地址，将会在虚拟机之间启用 IPv6 内部路由。默认启用 IPv4 内部路由。

DNS 域名: vnet1

取消(C) 后退(B) 完成(F)

图 13.4：创建虚拟网络

如果要创建转发的网络，请指定要将请求转发到的网络接口，以及以下其中一种转发模式：NAT（网络地址转换）模式会重新映射虚拟网络地址空间并允许共享单个 IP 地址，而路由模式则会将包从虚拟网络转发到 VM 主机服务器的物理网络，但不进行转换。

12. 如果您之前未指定 IPv6 网络地址空间定义，可以在虚拟机之间启用 IPv6 内部路由。
13. （可选）（可选）更改 DNS 域名。

14. 单击完成以创建新虚拟网络。VM 主机服务器上即会有一个新的虚拟网桥 `virbrX`，该网桥对应于新建的虚拟网络。您可以使用 `bridge link` 进行检查。`libvirt` 会自动添加 `iptables` 规则来允许传入/传出挂接到新 `virbrX` 设备的 Guest 的流量。

13.2.1.2 启动虚拟网络

要启动某个暂时停止的虚拟网络，请执行以下步骤：

1. 启动虚拟机管理器。在可用连接列表中，右键单击您需要为其配置虚拟网络的连接名称，然后选择细节。
2. 在连接细节窗口中，单击虚拟网络选项卡。您会看到可用于当前连接的所有虚拟网络的列表。
3. 要启动虚拟网络，请单击启动。

13.2.1.3 停止虚拟网络

要停止活动的虚拟网络，请执行以下步骤：

1. 启动虚拟机管理器。在可用连接列表中，右键单击您需要为其配置虚拟网络的连接名称，然后选择细节。
2. 在连接细节窗口中，单击虚拟网络选项卡。您会看到可用于当前连接的所有虚拟网络的列表。
3. 选择要停止的虚拟网络，然后单击停止。

13.2.1.4 删除虚拟网络

要删除 VM 主机服务器中的虚拟网络，请执行以下步骤：

1. 启动虚拟机管理器。在可用连接列表中，右键单击您需要为其配置虚拟网络的连接名称，然后选择细节。
2. 在连接细节窗口中，单击虚拟网络选项卡。您会看到可用于当前连接的所有虚拟网络的列表。

3. 选择要删除的虚拟网络，然后单击删除。

13.2.1.5 使用 **nsswitch** 获取 NAT 网络的 IP 地址（在 KVM 中）

- 在 VM 主机服务器上，安装用来为 libvirt 提供 NSS 支持的 libvirt-nss：

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

- 将 **libvirt** 添加到 **/etc/nsswitch.conf**：

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

- 如果 NSCD 正在运行，请将其重新启动：

```
tux > sudo systemctl restart nscd
```

现在，您可以从主机按名称访问 Guest 系统了。

NSS 模块的功能有限。它会读取 **/var/lib/libvirt/dnsmasq/*.status** 文件，以在描述 **dnsmasq** 所提供的每个租约的 JSON 记录中查找主机名和对应的 IP 地址。只能使用受 **dnsmasq** 支持且由 libvirt 管理的桥接网络在这些 VM 主机服务器上执行主机名转换。

有关更多信息，请参见http://wiki.libvirt.org/page/NSS_module。

13.2.2 使用 **virsh** 管理虚拟网络

您可以使用 **virsh** 命令行工具来管理 **libvirt** 提供的虚拟网络。要查看所有与网络相关的 **virsh** 命令，请运行以下命令

```
tux > sudo virsh help network
Networking (help keyword 'network'):
  net-autostart                autostart a network
  net-create                   create a network from an XML file
  net-define                   define (but don't start) a network from
  an XML file
```

<code>net-destroy</code>	destroy (stop) a network
<code>net-dumpxml</code>	network information in XML
<code>net-edit</code>	edit XML configuration for a network
<code>net-event</code>	Network Events
<code>net-info</code>	network information
<code>net-list</code>	list networks
<code>net-name</code>	convert a network UUID to network name
<code>net-start</code>	start a (previously defined) inactive network
<code>net-undefine</code>	undefine an inactive network
<code>net-update</code>	update parts of an existing network's configuration
<code>net-uuid</code>	convert a network name to network UUID

要查看特定 `virsh` 命令的简要帮助信息，请运行 `virsh help VIRSH_COMMAND`：

```
tux > sudo virsh help net-create
NAME
    net-create - create a network from an XML file

SYNOPSIS
    net-create <file>

DESCRIPTION
    Create a network.

OPTIONS
    [--file] <string>  file containing an XML network description
```

13.2.2.1 创建网络

要创建新的运行中虚拟网络，请运行以下命令

```
tux > sudo virsh net-create VNET_DEFINITION.xml
```

`VNET_DEFINITION.xml` XML 文件包含 `libvirt` 接受的虚拟网络的定义。

要定义新虚拟网络但不激活它，请运行以下命令

```
tux > sudo virsh net-define VNET_DEFINITION.xml
```

以下示例说明了不同类型的虚拟网络的定义。

例 13.1：基于 NAT 的网络

下面的配置可在 VM 主机服务器上无法使用传出连接时实现 VM Guest 传出连接。当没有 VM 主机服务器网络时，此配置可让 Guest 互相直接通讯。

```
<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="192.168.1.23.101"/> ❻
    <host mac="52:54:00:c7:92:db" name="host2.testing.com" \
      ip="192.168.1.23.102"/>
    <host mac="52:54:00:c7:92:dc" name="host3.testing.com" \
      ip="192.168.1.23.103"/>
  </dhcp>
</ip>
</network>
```

- ❶ 新虚拟网络的名称。
- ❷ 用于构造虚拟网络的网桥设备的名称。定义 <forward> 模式为 “nat” 或 “route” 的新网络（或者不包含 <forward> 元素的隔离网络）时，`libvirt` 将自动为网桥设备生成唯一的名称（如果未指定名称）。
- ❸ 包含 <forward> 元素表示该虚拟网络将连接到物理 LAN。`mode` 属性指定转发方法。最常用的模式为 “nat”（默认模式，表示网络地址转换）、“route”（直接转发到物理网络，不执行地址转换）和 “bridge”（在 `libvirt` 外部配置的网桥）。如果不指定 <forward> 元素，虚拟网络将与其他网络相隔离。有关转发模式的完整列表，请参见 <http://libvirt.org/formatnetwork.html#elementsConnect>。
- ❹ 网桥的 IP 地址和网络掩码。

- ⑤ 为虚拟网络启用 DHCP 服务器，并提供从指定的 `start` 到 `end` 属性范围内的 IP 地址。
- ⑥ 可选的 `<host>` 元素指定内置 DHCP 服务器要为其分配名称和预定义 IP 地址的主机。任何 IPv4 `host` 元素都必须指定以下设置：要为其指派给定名称的主机的 MAC 地址、要指派到该主机的 IP，以及 DHCP 服务器要为该主机分配的名称。IPv6 `host` 元素与 IPv4 略有不同，它没有 `mac` 属性，因为 MAC 地址在 IPv6 中没有明确的含义。IPv6 中使用 `name` 属性来标识要为其指派 IPv6 地址的主机。对于 DHCPv6，`name` 是由客户端发送到服务器的客户端主机的纯文本名称。请注意，您也可以使用这种指派特定 IP 地址的方法来代替 IPv4 中的 `mac` 属性。

例 13.2：路由网络

下面的配置会在不应用任何 NAT 的情况下将流量从虚拟网络路由到 LAN。必须在 VM 主机服务器网络上的路由器的路由表中预定义 IP 地址范围。

```
<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 Guest 流量只能通过 VM 主机服务器上的 `eth1` 网络设备传出。

例 13.3：隔离网络

此配置提供完全隔离的专用网络。Guest 可以相互通讯以及与 VM 主机服务器通讯，但无法访问 LAN 上的任何其他计算机，因为 XML 说明中没有 `<forward>` 元素。

```
<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 13.4：使用 VM 主机服务器上的现有网桥

此配置说明如何使用 VM 主机服务器的现有网桥 `br0`。VM Guest 直接连接到物理网络。其 IP 地址全部都在物理网络的子网中，并且传入和传出连接不存在任何限制。

```

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

```

13.2.2.2 列出网络

要列出 `libvirt` 可用的所有虚拟网络，请运行以下命令：

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

Name	State	Autostart	Persistent
crowbar	active	yes	yes
vnet_nated	active	yes	yes
vnet_routed	active	yes	yes
vnet_isolated	inactive	yes	yes

要列出可用域，请运行以下命令：

```
tux > sudo virsh list
```

Id	Name	State
1	nated_sles12sp3	running
...		

要获取运行中域的接口列表，请运行 `domifaddr DOMAIN`，或选择性地指定接口，以在输出中仅列出此接口。默认情况下，会额外输出接口的 IP 和 MAC 地址：

```
tux > sudo virsh domifaddr nated_sles12sp3 --interface vnet0 --source lease
```

Name	MAC address	Protocol	Address
vnet0	52:54:00:9e:0d:2b	ipv6	fd00:dead:beef:55::140/64
-	-	ipv4	192.168.100.168/24

要列显与指定域关联的所有虚拟接口的简要信息，请运行以下命令：

```
tux > sudo virsh domiflist nated_sles12sp3
```

Interface	Type	Source	Model	MAC
vnet0	network	vnet_nated	virtio	52:54:00:9e:0d:2b

13.2.2.3 获取有关网络的细节

要获取有关网络的详细信息，请运行以下命令：

```
tux > sudo virsh net-info vnet_routed
```

```
Name:          vnet_routed
UUID:          756b48ff-d0c6-4c0a-804c-86c4c832a498
Active:        yes
Persistent:    yes
Autostart:     yes
Bridge:        virbr5
```

13.2.2.4 启动网络

要启动某个已定义的非活动网络，请使用以下命令查找其名称（或唯一标识符，即 UUID）：

```
tux > sudo virsh net-list --inactive
```

Name	State	Autostart	Persistent
vnet_isolated	inactive	yes	yes

然后运行：

```
tux > sudo virsh net-start vnet_isolated
```



```
Network vnet_isolated started
```

13.2.2.5 停止网络

要停止某个活动的网络，请使用以下命令查找其名称（或唯一标识符，即 UUID）：

```
tux > sudo virsh net-list --inactive
```

Name	State	Autostart	Persistent

vnet_isolated	active	yes	yes

然后运行：

```
tux > sudo virsh net-destroy vnet_isolated  
Network vnet_isolated destroyed
```

13.2.2.6 去除网络

要从 VM 主机服务器中永久去除某个非活动网络的定义，请运行以下命令：

```
tux > sudo virsh net-undefine vnet_isolated  
Network vnet_isolated has been undefined
```

14 使用虚拟机管理器配置虚拟机

虚拟机管理器的细节视图提供有关 VM Guest 完整配置和硬件要求的深入信息。使用此视图还可以更改 Guest 配置，或者添加和修改虚拟硬件。要访问此视图，请在虚拟机管理器中打开 Guest 的控制台，然后从菜单中选择视图 > 细节，或者在工具栏中单击显示虚拟硬件细节。

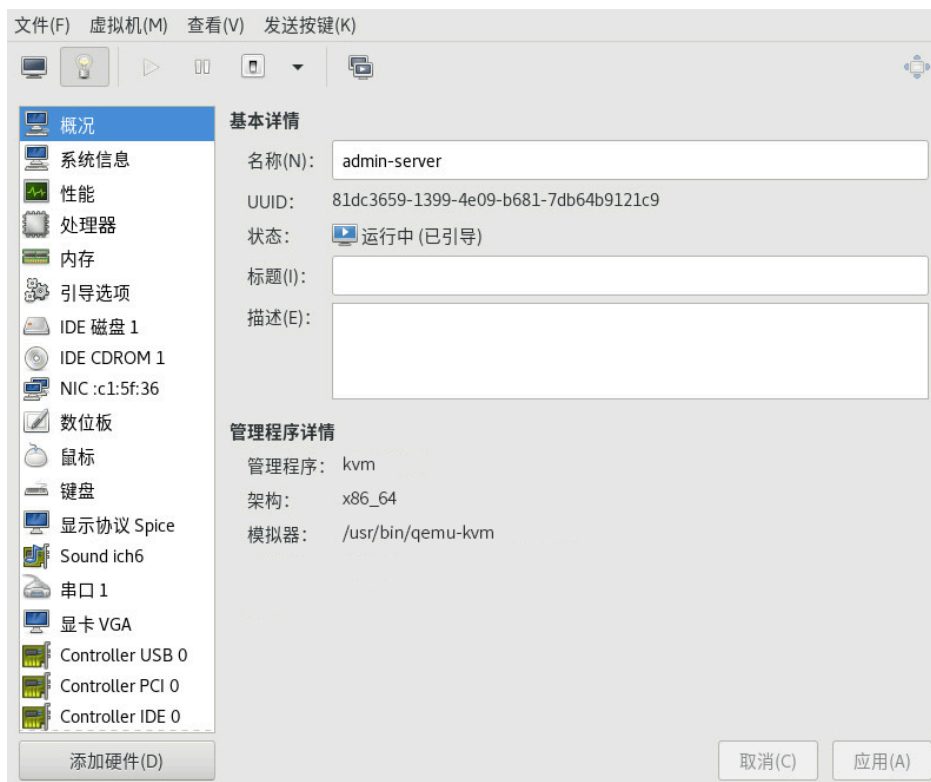


图 14.1：VM GUEST 的细节视图

窗口的左侧面板列出了 VM Guest 概述及已安装的硬件。在列表中单击某个项后，可以在细节视图中访问其详细设置。您可以根据需要更改硬件参数，然后单击应用确认更改。有些更改会立即生效，而有些更改需要重引导计算机，`virt-manager` 会预先通知您此情况。

要从 VM Guest 中去除安装的硬件，请在左侧面板中选择相应的列表项，然后单击窗口右下方的去除。

要添加新硬件，请单击左侧面板下方的添加硬件，然后在添加新虚拟硬件窗口中选择要添加的硬件类型。修改其参数并单击完成进行确认。

下列章节介绍要添加的特定硬件类型的配置选项。其中不会重点介绍如何修改现有的硬件，因为选项均相同。

14.1 计算机设置

本节介绍虚拟化处理器和内存硬件的设置。这些组件对于 VM Guest 至关重要，因此不能将其去除。本节还将介绍如何查看概述和性能信息，以及如何更改引导参数。

14.1.1 概览

概览显示有关 VM Guest 和超级管理程序的基本细节。

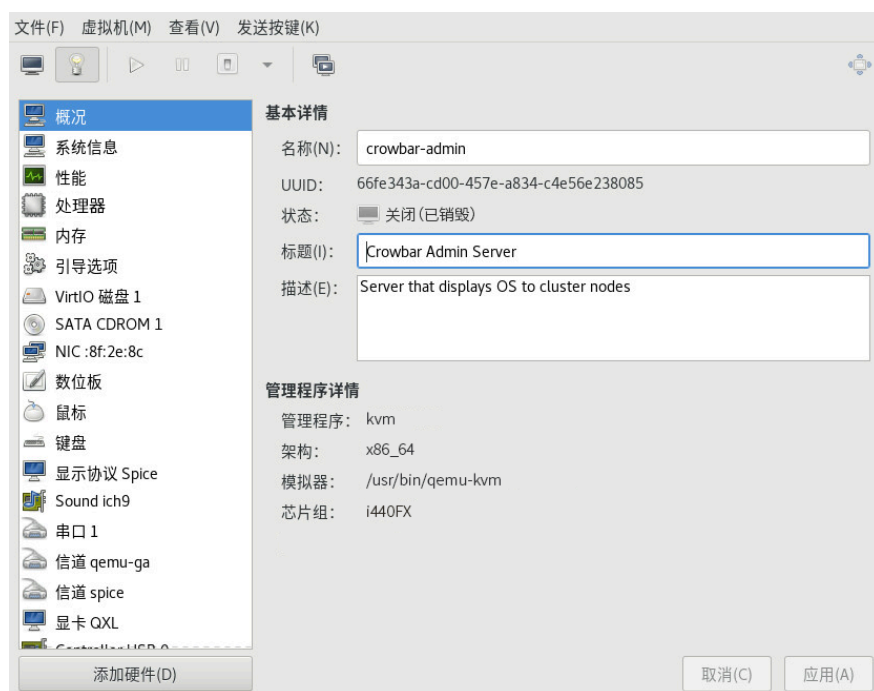


图 14.2：概览细节

名称、标题和说明均可编辑，有助于您在虚拟机管理器的计算机列表中识别 VM Guest。

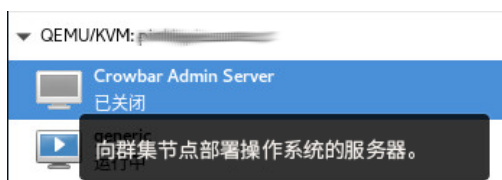


图 14.3：VM GUEST 标题和说明

UUID 显示虚拟机的全局唯一标识符，而状态显示虚拟机的当前状态 — 正在运行、已暂停或已关闭。

超级管理程序细节部分显示超级管理程序类型、CPU 体系结构、使用的模拟器和芯片组类型。所有超级管理程序参数都不可更改。

14.1.2 性能

性能显示 CPU 和内存使用率以及磁盘和网络 I/O 的定期更新图表。

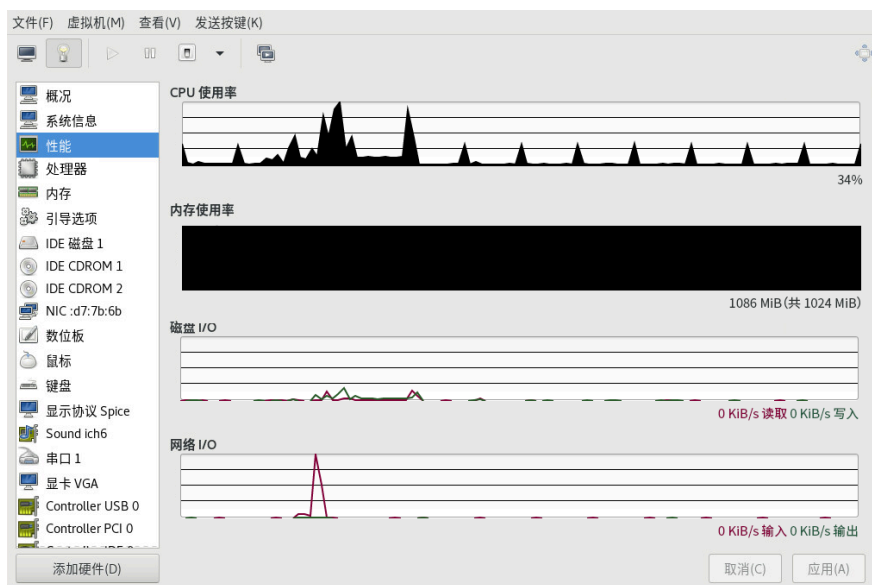


图 14.4：性能



提示：启用已禁用的图表

图表视图中的所有图表并非默认都已启用。要启用这些图表，请转到文件 > 视图管理器，然后选择编辑 > 首选项 > 轮询，检查您要查看的图表是否定期更新。

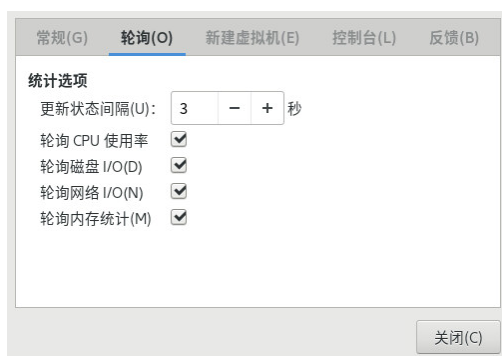


图 14.5：统计图

14.1.3 处理器

处理器视图包含有关 VM Guest 处理器配置的详细信息。

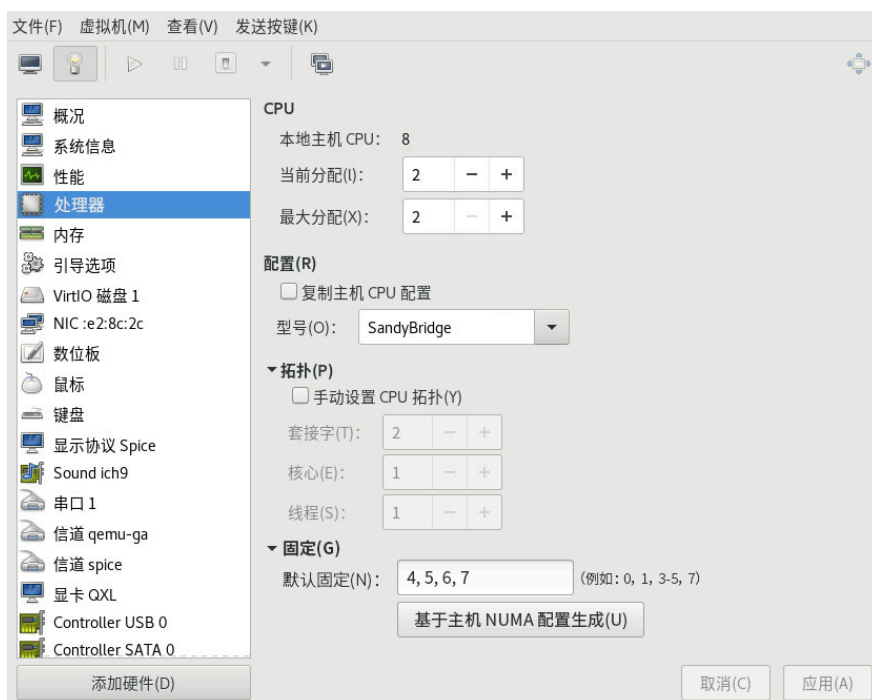


图 14.6：处理器视图

在 CPU 部分，可以配置与分配的 CPU 数量相关的多个参数。

本地主机 CPU

VM 主机服务器上实际安装的 CPU 数量。

当前分配

当前分配的 CPU 数量。可以通过增大此值（但不超过最大分配量）来热插入更多 CPU。

最大分配

可为当前会话分配的最大 CPU 数量。对此值所做的任何更改将在 VM Guest 下次重引导后生效。

配置部分可让您配置 CPU 型号和拓扑。

如果激活复制主机 CPU 配置选项，将为 VM Guest 使用主机 CPU 型号。否则，您将需要通过下拉框指定 CPU 型号。

激活手动设置 CPU 拓扑后，可以指定 CPU 的自定义插槽数、核心数和线程数。

14.1.4 内存

内存视图包含有关 VM Guest 可用内存的信息。

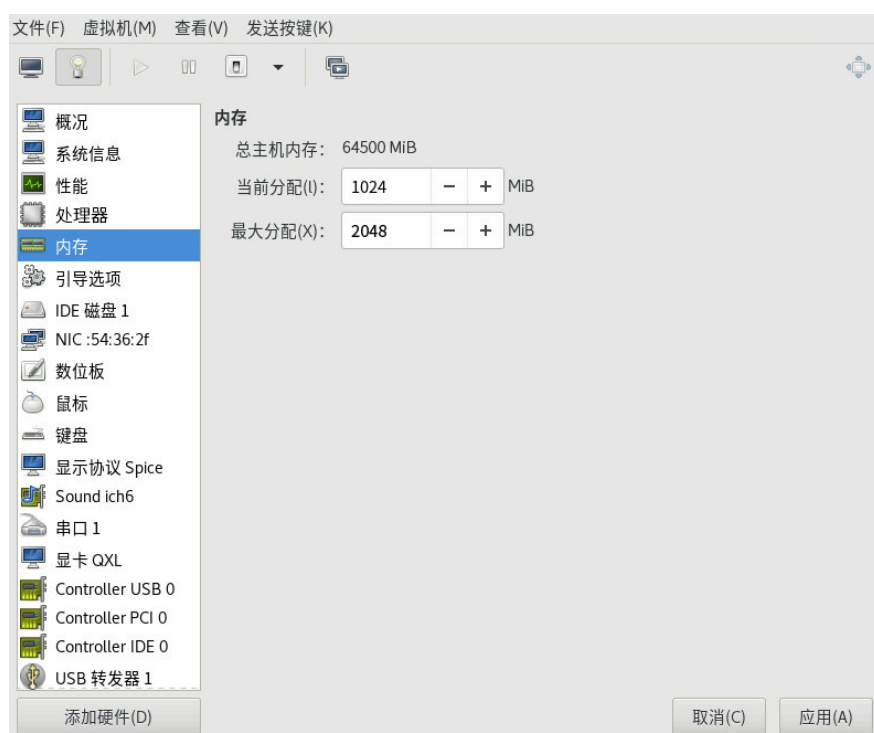


图 14.7：内存视图

总主机内存

VM 主机服务器上安装的内存总量。

当前分配

当前可供 VM Guest 使用的内存量。可以通过增大此值（但不超过最大分配数量）来热插入更多内存。

最大分配

可热插入的当前可用内存的最大值。对此值所做的任何更改将在 VM Guest 下次重引导后生效。

14.1.5 引导选项

引导选项引入了影响 VM Guest 引导进程的选项。

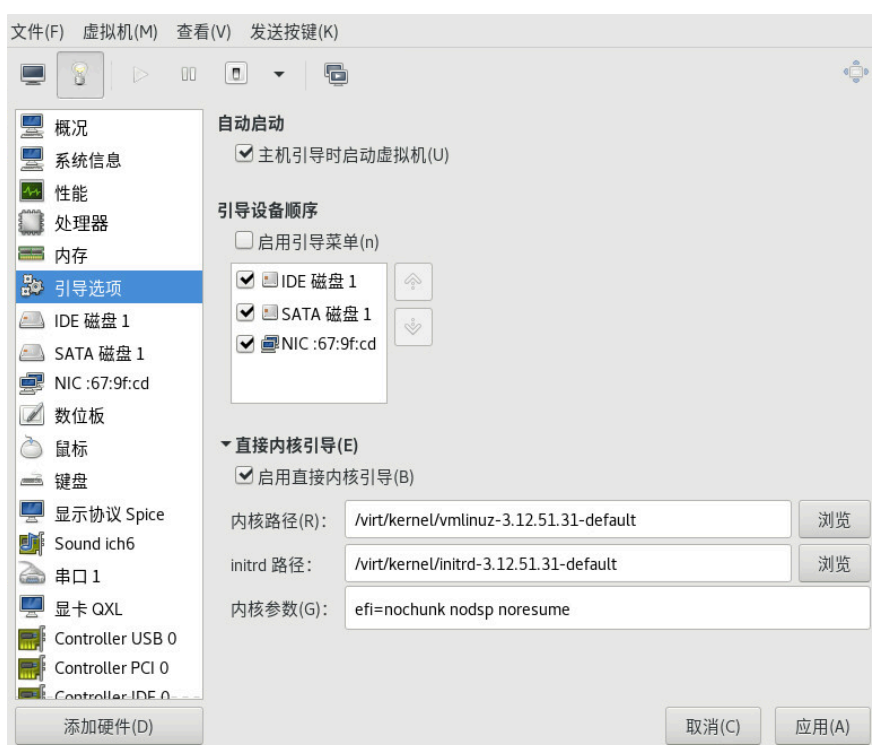


图 14.8：引导选项

在自动启动部分，可以指定是否应在 VM 主机服务器引导阶段自动启动虚拟机。

在引导设备顺序中，激活用于引导 VM Guest 的设备。可以使用列表右侧的向上和向下箭头按钮更改其顺序。要在 VM Guest 启动时从可引导设备列表中进行选择，请激活启用引导菜单。

要引导其他内核而不是引导设备上的内核，请激活启用直接内核引导，并指定替代内核以及位于 VM 主机服务器文件系统上的 initrd 的路径。您还可以指定要传递给所装载内核的内核参数。

14.2 储存

本节提供储存设备配置选项的详细说明。其中包括硬盘和可移动媒体，例如 USB 或 CD-ROM 驱动器。

过程 14.1：添加新储存设备

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择储存。

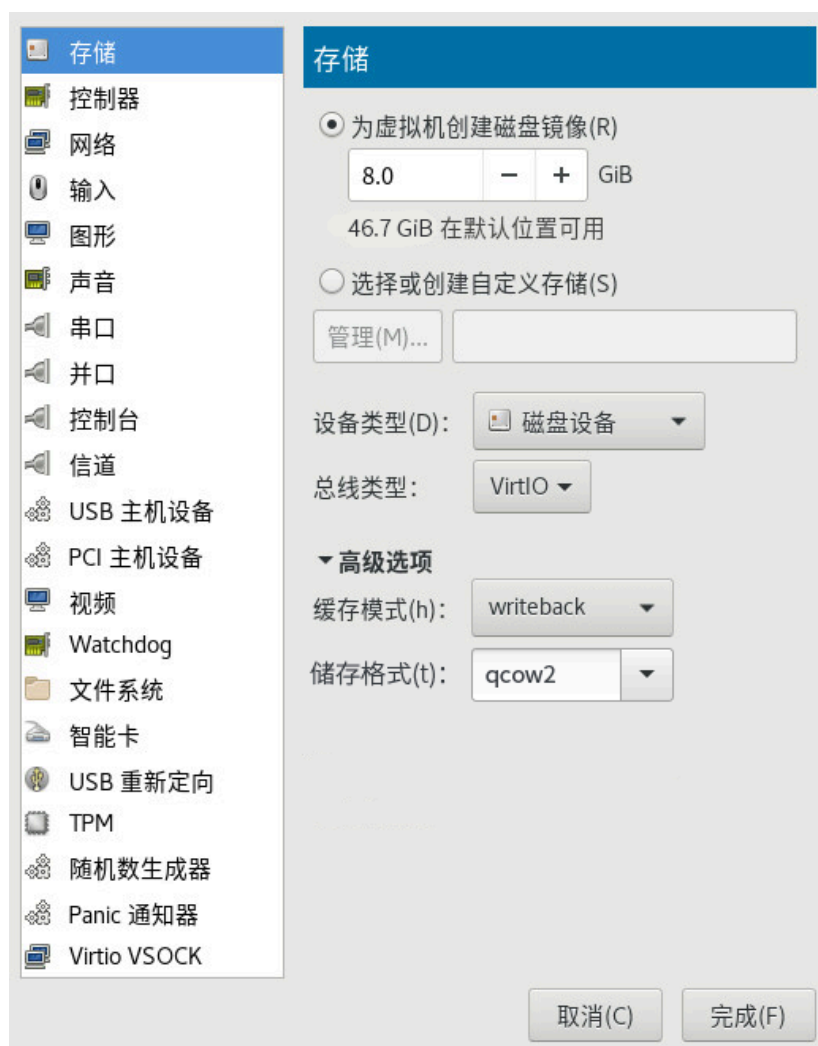


图 14.9：添加新储存

2. 要在默认位置创建 qcow2 磁盘映像，请激活创建虚拟机的磁盘映像，然后以 GB 为单位指定其大小。

要以更高的控制度创建磁盘映像，请激活选择或创建自定义储存，然后单击管理以管理储存池和映像。选择储存卷窗口即会打开，其中提供的功能与第 12.1 节 “使用虚拟机管理器管理储存” 中所述的储存选项卡基本相同。



提示：支持的储存格式

SUSE 仅支持以下储存格式：raw 和 qcow2。

3. 在成功创建并指定磁盘映像文件后，请指定设备类型。设备类型可为以下选项之一：

- 磁盘设备
- CDROM 设备：不允许使用创建虚拟机的磁盘映像。
- 软盘设备：不允许使用创建虚拟机的磁盘映像。
- LUN 直通：如果想直接使用现有的 SCSI 储存而不将其添加到储存池，则需使用此选项。

4. 选择设备的总线类型。可用选项的列表取决于您在上一步中选择的设备类型。基于 VirtIO 的类型使用半虚拟化驱动程序。

5. 在高级选项部分选择首选的缓存模式。有关缓存模式的详细信息，请参见第 17 章 “磁盘缓存模式”。

6. 单击完成确认您的设置。新储存设备随即显示在左侧面板中。

14.3 控制器

本节重点介绍如何添加和配置新控制器。

过程 14.2：添加新控制器

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择控制器。

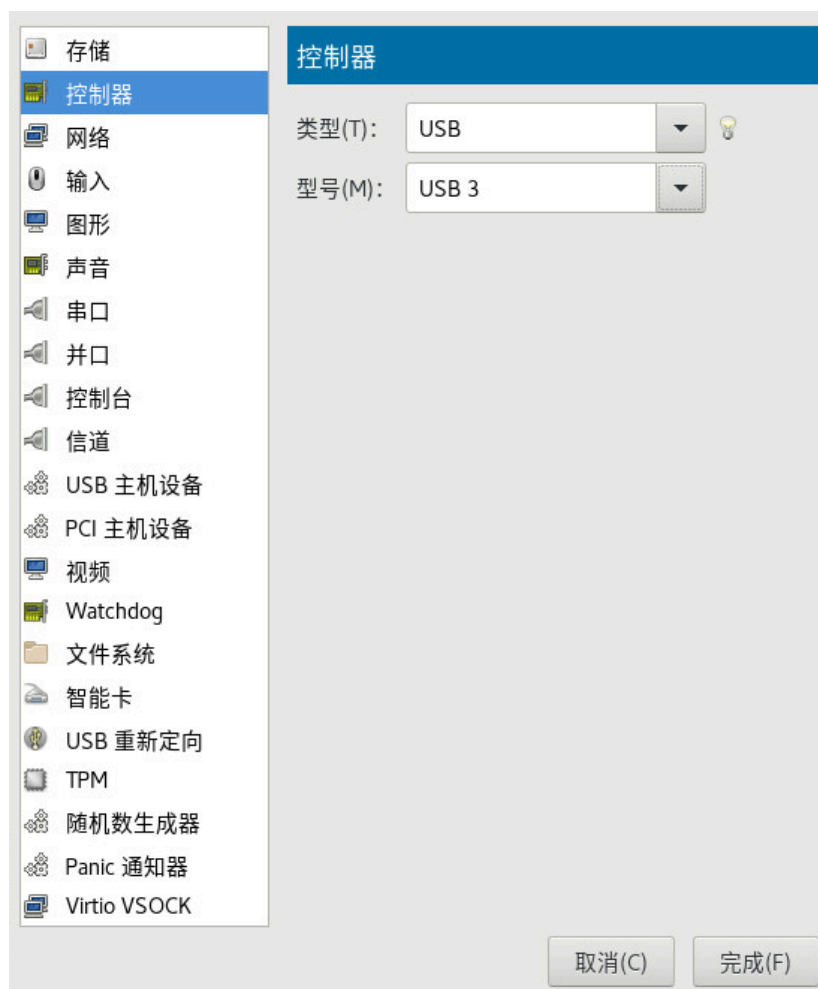


图 14.10：添加新控制器

2. 选择控制器的类型。可以选择 IDE、软盘、SCSI、SATA、VirtIO Serial（半虚拟化）、USB 或 CCID（智能卡设备）。
3. （可选）如果选择了 USB 或 SCSI 控制器，请选择控制器型号。
4. 单击完成确认您的设置。新控制器随即显示在左侧面板中。

14.4 网络

本节介绍如何添加和配置新网络设备。

过程 14.3：添加新网络设备

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择网络。

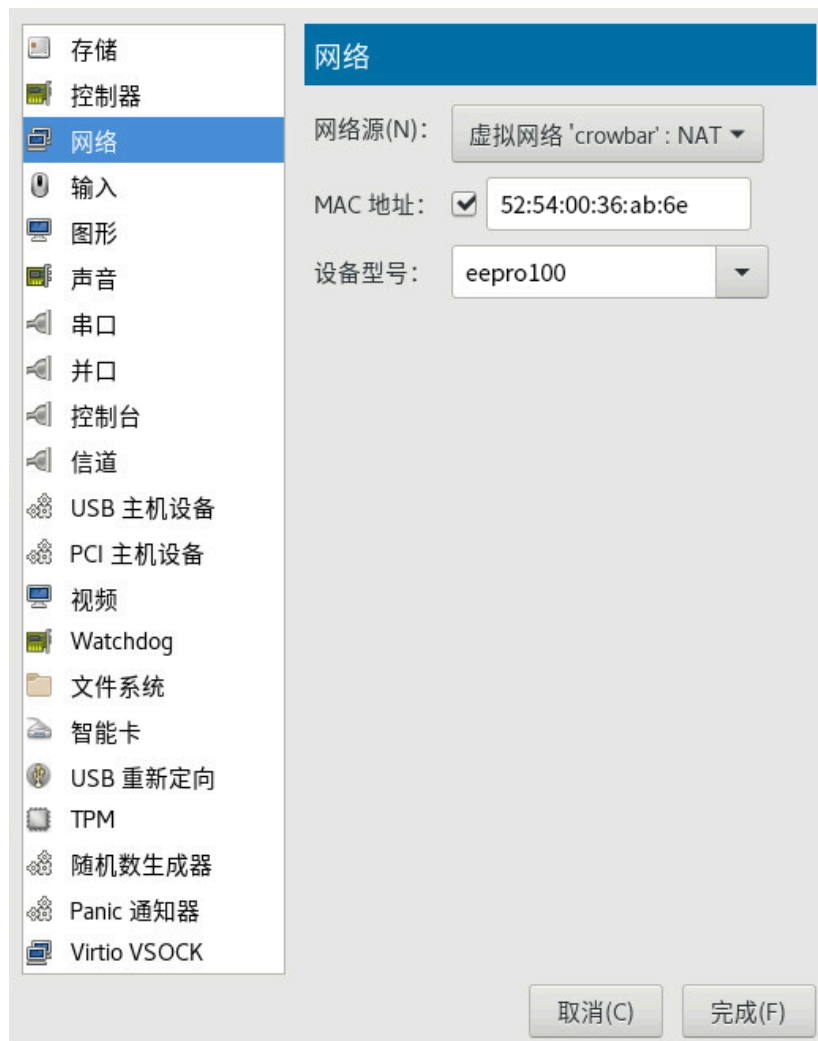


图 14.11：添加新控制器

2. 在网络源列表中，选择网络连接的来源。该列表包含 VM 主机服务器的可用物理网络接口、网桥或网络绑定。您还可以将 VM Guest 指派到已定义的虚拟网络。有关使用虚拟机管理器设置虚拟网络的详细信息，请参见第 13 章“管理网络”。
3. 指定网络设备的 MAC 地址。尽管出于方便虚拟机管理器会预先填充一个随机值，但我们建议提供适合您网络环境的 MAC 地址，以免发生网络冲突。
4. 从列表中选择设备型号。可以保留超级管理程序默认值，或者指定 e1000、rtl8139 或 virtio 型号中的一个。请注意，virtio 使用半虚拟化设备。

5. 单击完成确认您的设置。新网络设备随即显示在左侧面板中。

14.5 输入设备

本节重点介绍如何添加和配置新输入设备，例如鼠标、键盘或绘图板。

过程 14.4：添加新输入设备

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择输入。

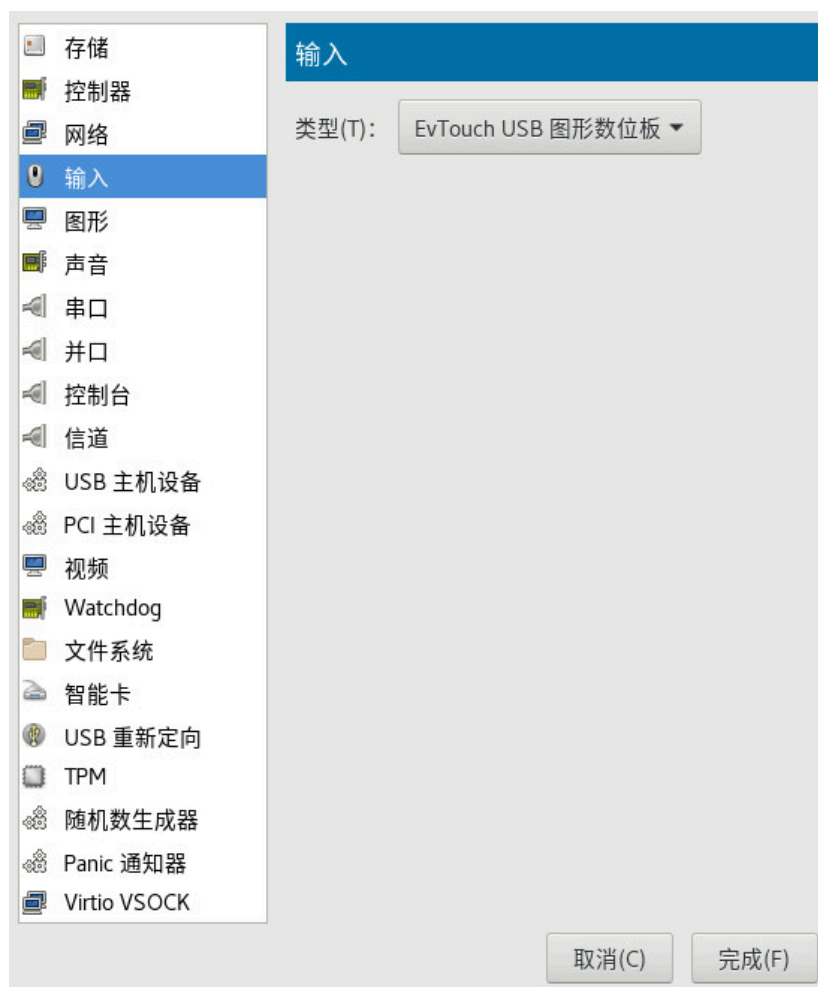


图 14.12：添加新输入设备

2. 从列表中选择设备类型。
3. 单击完成确认您的设置。新输入设备随即显示在左侧面板中。



提示：启用无缝且同步的鼠标指针移动

当您使用鼠标在 VM Guest 的控制台中单击时，指针将由控制台窗口捕获，除非显式释放指针（按 **Alt + Ctrl**），否则无法在控制台外部使用指针。为了防止控制台独占按键，以及在主机与 Guest 之间启用无缝指针移动，请按照[过程 14.4 “添加新输入设备”](#)中的说明将一个 EvTouch USB 绘图板添加到 VM Guest。

添加绘图板的另一个好处是，在 Guest 上使用图形环境时可以同步 VM 主机服务器与 VM Guest 之间的鼠标指针移动。如果不在 Guest 上配置绘图板，您经常会看到两个有拖尾现象（一个指针拖在另一个指针后面）的指针。

14.6 视频

本节介绍如何添加和配置新视频设备。

过程 14.5：添加视频设备

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择视频。

2.

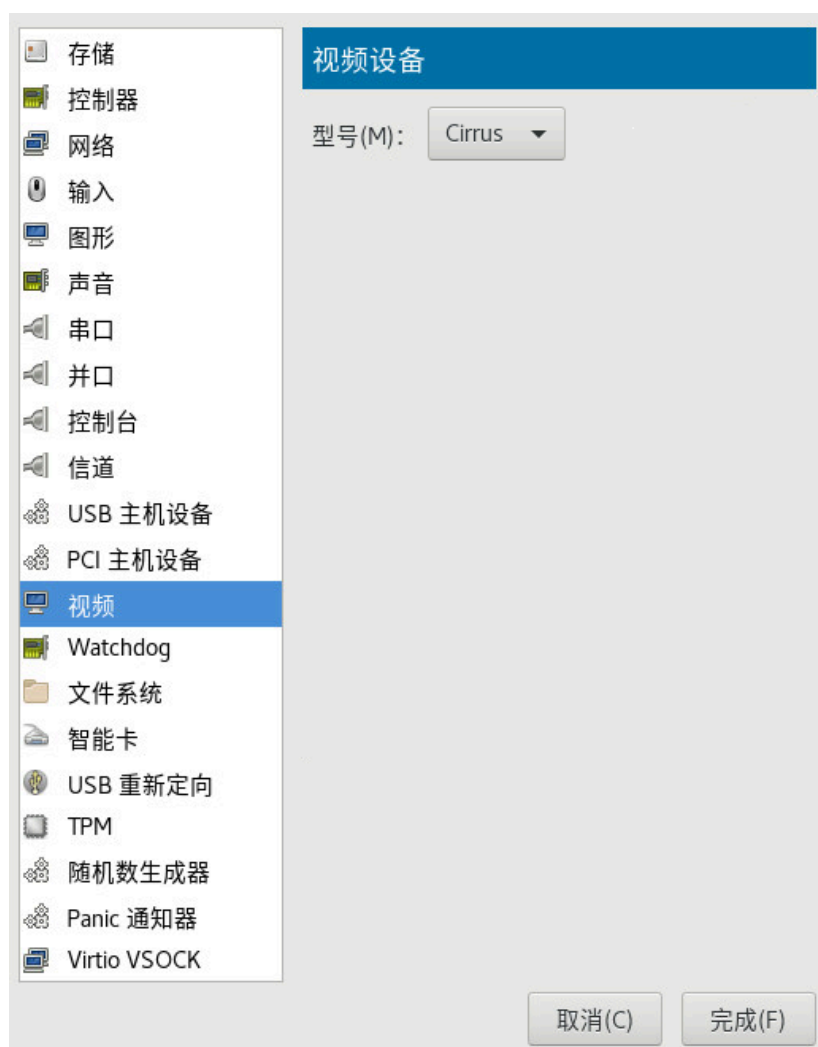


图 14.13：添加新视频设备

3. 从列表中选择型号。可供选择的型号如下：

- Cirrus
- QXL
- VGA
- Virtio
- VMVGA
- Xen



注意：次要视频设备

只能将 QXL 和 Virtio 添加为次要视频设备。

4. 单击完成确认您的设置。新视频设备随即显示在左侧面板中。

14.7 USB 重定向器

可以使用 USB 重定向器将连接到客户端计算机的 USB 设备重定向到 VM Guest。

过程 14.6：添加 USB 重定向器

1. 在左侧面板下方，单击添加硬件打开添加新虚拟硬件窗口。在此窗口中选择 USB 重新定向。

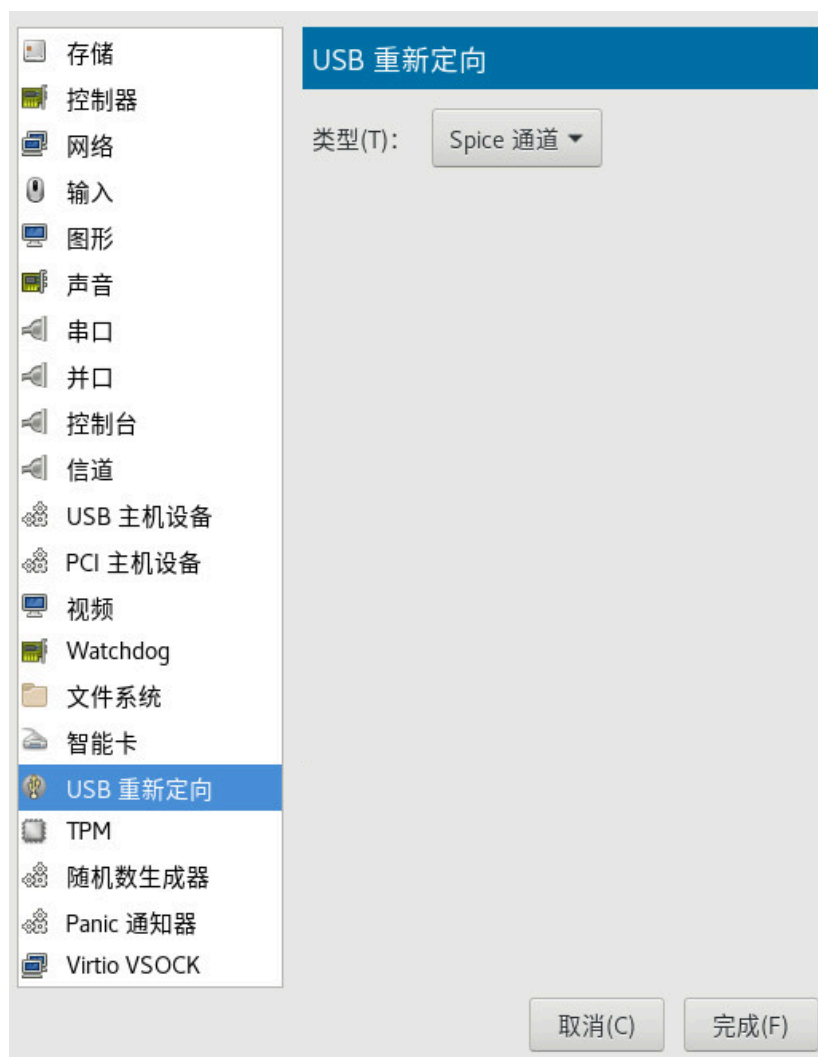


图 14.14：添加新 USB 重定向器

2. 从列表中选择设备类型。可以选择 Spice Channel 或 TCP 重定向器。
3. 单击完成确认您的设置。新 USB 重定向器随即显示在左侧面板中。

14.8 杂项

智能卡

可以通过 Smartcard 元素添加智能卡功能。然后，物理 USB 智能卡读卡器便可以直通到 VM Guest。

看门狗

系统还支持虚拟看门狗设备。可通过 Watchdog 元素创建这些设备。可以指定型号以及设备操作。



提示：虚拟看门狗设备的要求

要使用 QA 虚拟看门狗设备，需要在 VM Guest 中安装特定的驱动程序和守护程序，否则虚拟看门狗设备将无法正常工作。

TPM

可以通过 TPM 元素添加 TPM 功能，以在 VM Guest 中使用主机 TPM 设备。



提示：虚拟 TPM

每次只能在一个 VM Guest 中使用主机 TPM。

14.9 使用虚拟机管理器添加 CD/DVD-ROM 设备

KVM 通过直接访问 VM 主机服务器上的物理驱动器或访问 ISO 映像来支持 VM Guest 中的 CD 和 DVD-ROM。要基于现有 CD 或 DVD 创建 ISO 映像，请使用 **dd**：

```
tux > sudo dd if=/dev/CD_DVD_DEVICE of=my_distro.iso bs=2048
```

要将 CD/DVD-ROM 设备添加到 VM Guest，请执行以下操作：

1. 双击虚拟机管理器中的某个 VM Guest 项打开其控制台，然后选择视图 > 细节切换到细节视图。
2. 单击添加硬件并在弹出窗口中选择储存。
3. 将设备类型更改为 IDE CDROM。
4. 选择选择或创建自定义储存。

- a. 要将设备指派到物理媒体，请在管理旁边输入 VM 主机服务器 CD/DVD-ROM 设备的路径（例如 `/dev/cdrom`）。或者，使用管理打开文件浏览器，然后单击浏览本地选择设备。仅当 VM 主机服务器上已启动虚拟机管理器时，才能将设备指派到物理媒体。
 - b. 要将设备指派到现有映像，请单击管理以从储存池中选择映像。如果 VM 主机服务器上已启动虚拟机管理器，您也可以单击浏览本地从文件系统上的另一个位置选择映像。选择某个映像并单击选择卷以关闭文件浏览器。
5. 单击完成以保存新虚拟化设备。
 6. 重引导 VM Guest 以使新设备可用。有关更多信息，请参见第 14.11 节 “使用虚拟机管理器弹出和更换软盘或 CD/DVD-ROM 媒体”。

14.10 使用虚拟机管理器添加软盘设备

KVM 目前仅支持使用软盘映像 — 不支持使用物理软盘驱动器。使用 `dd` 基于现有软盘创建软盘映像：

```
tux > sudo dd if=/dev/fd0 of=/var/lib/libvirt/images/floppy.img
```

要创建空软盘映像，请使用以下命令之一：

Raw 映像

```
tux > sudo dd if=/dev/zero of=/var/lib/libvirt/images/floppy.img bs=512  
count=2880
```

FAT 格式映像

```
tux > sudo mkfs.msdos -C /var/lib/libvirt/images/floppy.img 1440
```

要将软盘设备添加到 VM Guest，请执行以下操作：

1. 双击虚拟机管理器中的某个 VM Guest 项打开其控制台，然后选择视图 > 细节切换到细节视图。

2. 单击添加硬件并在弹出窗口中选择储存。
3. 将设备类型更改为软盘。
4. 选择选择或创建自定义储存，然后单击管理以从储存池中选择一个现有映像。如果 VM 主机服务器上已启动虚拟机管理器，您也可以单击浏览本地从文件系统上的另一个位置选择映像。选择某个映像并单击选择卷以关闭文件浏览器。
5. 单击完成以保存新虚拟化设备。
6. 重引导 VM Guest 以使新设备可用。有关更多信息，请参见第 14.11 节“使用虚拟机管理器弹出和更换软盘或 CD/DVD-ROM 媒体”。

14.11 使用虚拟机管理器弹出和更换软盘或 CD/DVD-ROM 媒体

无论您使用的是 VM 主机服务器的物理 CD/DVD-ROM 设备，还是 ISO/软盘映像，在更换 VM Guest 中现有设备的媒体或映像之前，都需要先将媒体与 Guest 断开连接。

1. 双击虚拟机管理器中的某个 VM Guest 项打开其控制台，然后选择视图 > 细节切换到细节视图。
2. 选择软盘或 CD/DVD-ROM 设备，然后单击断开连接以“弹出”媒体。
3. 要“插入”新媒体，请单击连接。
 - a. 如果使用的是 VM 主机服务器的物理 CD/DVD-ROM 设备，请先更换设备中的媒体（这可能需要先在 VM 主机服务器上卸载该媒体，然后再将其弹出）。然后选择 CD-ROM 或 DVD，并从下拉框中选择设备。
 - b. 如果您使用的是 ISO 映像，请选择 ISO 映像位置，然后单击管理以选择映像。从远程主机连接时，只能选择现有储存池中的映像。
4. 单击确定以完成操作。现在便可在 VM Guest 中访问新媒体了。

14.12 将主机 PCI 设备指派到 VM Guest

可以直接将主机 PCI 设备指派到 Guest（PCI 直通）。将 PCI 设备指派到某个 VM Guest 后，除非重新指派，否则在主机上无法使用该设备，其他 VM Guest 也不能使用该设备。此功能的先决条件是 VM 主机服务器配置符合[重要：VFIO 和 SR-IOV 的要求](#)中所述的要求。

14.12.1 使用虚拟机管理器添加 PCI 设备

以下过程说明如何使用虚拟机管理器将 PCI 设备添加到 VM Guest：

1. 双击虚拟机管理器中的某个 VM Guest 项打开其控制台，然后选择视图 > 细节切换到细节视图。
2. 单击添加硬件，然后在左侧面板中选择 PCI 主机设备类别。窗口右侧将显示可用 PCI 设备的列表。

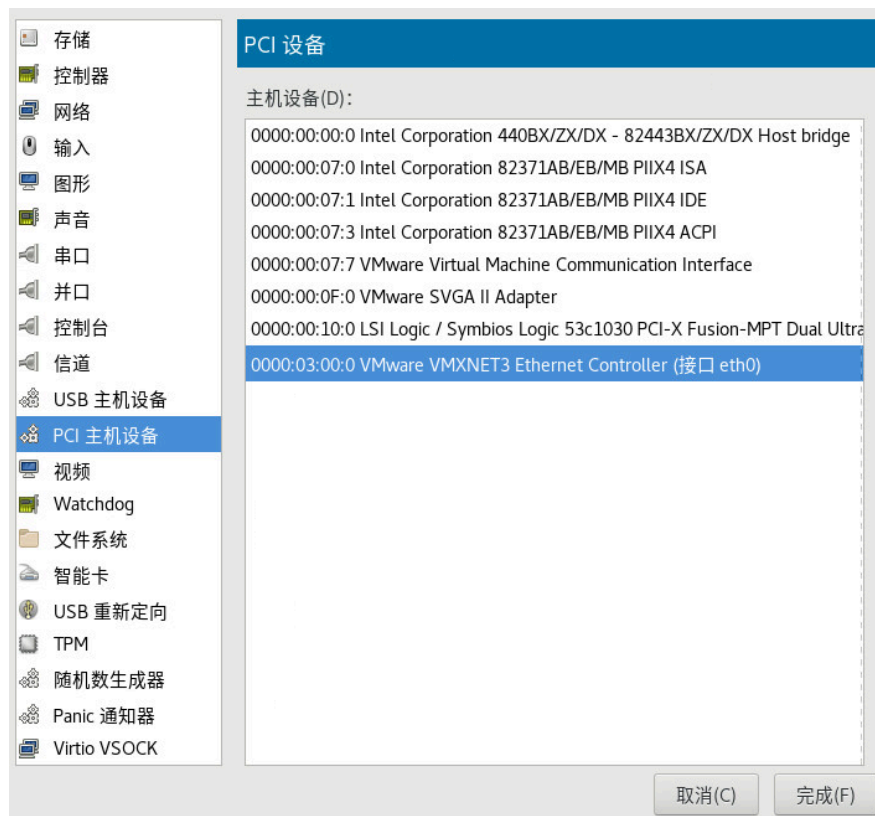


图 14.15：添加 PCI 设备

3. 在可用 PCI 设备的列表中，选择您要传递给 Guest 的设备。单击完成以确认。



提示：指派 PCI 设备需要关闭 VM Guest

尽管可按上文所述将 PCI 设备指派到运行中的 VM Guest，但只有在关闭再重引导 VM Guest 之后，该设备才可用。

14.13 将主机 USB 设备指派到 VM Guest

与指派主机 PCI 设备（请参见第 14.12 节“将主机 PCI 设备指派到 VM Guest”）类似，您可以直接将主机 USB 设备指派到 Guest。将 USB 设备指派到某个 VM Guest 后，除非重新指派，否则在主机上无法使用该设备，其他 VM Guest 也不能使用该设备。

14.13.1 使用虚拟机管理器添加 USB 设备

要使用虚拟机管理器将主机 USB 设备指派到 VM Guest，请执行以下步骤：

1. 双击虚拟机管理器中的某个 VM Guest 项打开其控制台，然后选择视图 > 细节切换到细节视图。
2. 单击添加硬件，然后在左侧面板中选择 USB 主机设备类别。窗口右侧将显示可用 USB 设备的列表。

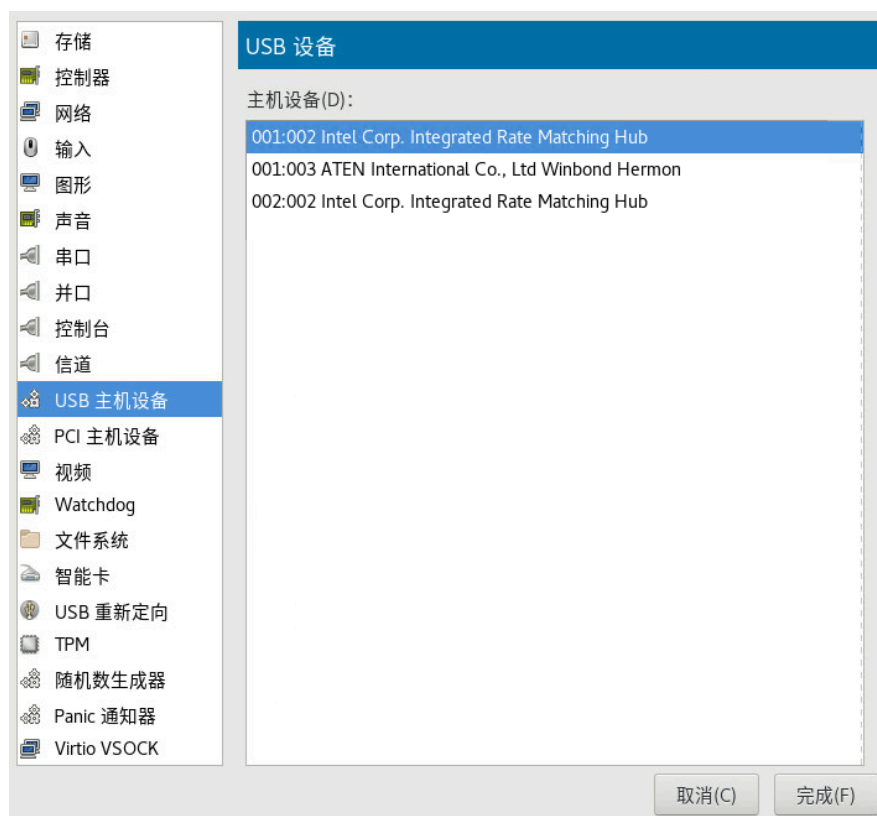


图 14.16：添加 USB 设备

3. 在可用 USB 设备的列表中，选择您要传递给 Guest 的设备。单击完成以确认。新 USB 设备随即显示在细节视图的左侧窗格中。



提示：去除 USB 设备

要去除主机 USB 设备指派，请在细节视图的左侧窗格中单击此设备，然后单击去除以确认。

15 使用 **virsh** 配置虚拟机

您也可以在命令行上使用 **virsh** 来配置虚拟机 (VM)，以此替代虚拟机管理器。使用 **virsh** 可以控制 VM 的状态、编辑 VM 的配置，甚至将 VM 迁移到另一台主机。下列章节介绍如何使用 **virsh** 来管理 VM。

15.1 编辑 VM 配置

VM 配置储存在 `/etc/libvirt/qemu/` 中的某个 XML 文件内，其内容类似如下示例：

例 15.1：示例 XML 配置文件

```
<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-i440fx-2.11'>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>
```

如果您要编辑 VM Guest 的配置，请检查它是否处于脱机状态：

```
tux > sudo virsh list --inactive
```

如果您的 VM Guest 在此列表中，则表明您可以放心地编辑其配置：

```
tux > sudo virsh edit NAME_OF_VM_GUEST
```

在保存更改之前，**virsh** 会根据 RelaxNG 纲要验证您的输入。

15.2 更改计算机类型

使用 **virt-install** 工具安装时，VM Guest 的计算机类型默认为 `pc-i440fx`。计算机类型储存在 VM Guest 配置文件中的 `type` 元素内：

```
<type arch='x86_64' machine='pc-i440fx-2.3'>hvm</type>
```

下面的过程示范了如何将此值更改为 `q35` 计算机类型。值 `q35` 表示一种 Intel* 芯片组，其中包括 **PCIe**，最多支持 12 个 USB 端口，并支持 **SATA** 和 **IOMMU**。

过程 15.1：更改计算机类型

1. 检查您的 VM Guest 是否处于非活动状态：

```
tux > sudo virsh list --inactive
```

Id	Name	State
-	sles15	shut off

2. 编辑此 VM Guest 的配置：

```
tux > sudo virsh edit sles15
```


3. 将 `machine` 属性的值替换为 `pc-q35-2.0`：

```
<type arch='x86_64' machine='pc-q35-2.0'>hvm</type>
```

4. 重启 VM Guest：

```
tux > sudo virsh start sles15
```

5. 检查计算机类型是否已更改。登录到 VM Guest 并运行以下命令：

```
tux > sudo dmidecode | grep Product
Product Name: Standard PC (Q35 + ICH9, 2009)
```



提示：计算机类型更新建议

每当升级主机系统上的 QEMU 版本时（例如，将 VM 主机服务器升级到新服务包时），请将 VM Guest 的计算机类型升级到最新的可用版本。要进行检查，请在 VM 主机服务器上使用 `qemu-system-x86_64 -M help` 命令。

默认计算机类型（例如 `pc-i440fx`）会定期更新。如果您的 VM Guest 仍在 `pc-i440fx-1.X` 计算机类型上运行，我们强烈建议更新到 `pc-i440fx-2.X`。这样就可以利用计算机定义中最近的更新和更正，并确保将来可以更好地兼容。

15.3 配置超级管理程序功能

`libvirt` 可自动启用一组默认的超级管理程序功能（这些功能在大多数情况下已够用），同时还允许按需启用和禁用功能。例如，Xen 不支持默认启用 PCI 直通。必须使用 [直通](#) 设置来启用此功能。可以使用 `virsh` 来配置超级管理程序功能。请查看 VM Guest 配置文件中的 `<features>` 元素，并根据需要调整各项功能。沿用 Xen 直通示例：

```
tux > sudo virsh edit sle15sp1
<features>
  <xen>
    <passthrough/>
  </xen>
</features>
```

保存更改并重新启动 VM Guest。

有关详细信息，请参见 <https://libvirt.org/formatdomain.html#elementsFeatures> 上 libvirt 的《Domain XML format》（域 XML 格式）手册中的“Hypervisor features”（超级管理程序功能）一节。

15.4 配置 CPU 分配

分配的 CPU 数量储存在 `/etc/libvirt/qemu/` 下 VM Guest XML 配置文件中的 `vcpu` 元素内：

```
<vcpu placement='static'>1</vcpu>
```

在此示例中，只为 VM Guest 分配了一个 CPU。下面的过程说明如何更改分配给 VM Guest 的 CPU 数量：

1. 检查您的 VM Guest 是否处于非活动状态：

```
tux > sudo virsh list --inactive
```

Id	Name	State
-	sles15	shut off

2. 编辑现有 VM Guest 的配置：

```
tux > sudo virsh edit sles15
```

3. 更改分配的 CPU 数量：

```
<vcpu placement='static'>2</vcpu>
```

4. 重新启动 VM Guest：

```
tux > sudo virsh start sles15
```

5. 检查 VM 中的 CPU 数量是否已更改。

```
tux > 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
```

15.5 更改引导选项

VM Guest 的引导菜单包含在 `os` 元素中，其内容通常如下所示：

```
<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>
```

此示例中有两个可用设备：`hd` 和 `cdrom`。配置还会反映实际引导顺序，因此 `cdrom` 先于 `hd` 引导。

15.5.1 更改引导顺序

VM Guest 的引导顺序通过 XML 配置文件中的设备顺序来表示。由于设备可以互换，因此可以更改 VM Guest 的引导顺序。

1. 打开 VM Guest 的 XML 配置。

```
tux > sudo virsh edit sles15
```

2. 更改可引导设备的顺序。

```
...  
<boot dev='cdrom' />  
<boot dev='hd' />  
...
```

3. 通过查看 VM Guest 的 BIOS 中的引导菜单来检查引导顺序是否已更改。

15.5.2 使用直接内核引导

使用直接内核引导可以从主机上储存的内核和 initrd 引导。在 `kernel` 和 `initrd` 元素中设置这两个文件的路径：

```
<os>  
...  
<kernel>/root/f8-i386-vmlinuz</kernel>  
<initrd>/root/f8-i386-initrd</initrd>  
...  
</os>
```

要启用直接内核引导，请执行以下操作：

1. 打开 VM Guest 的 XML 配置：

```
tux > sudo virsh edit sles15
```

2. 在 `os` 元素中，添加 `kernel` 元素以及主机上内核文件的路径：

```
...  
<kernel>/root/f8-i386-vmlinuz</kernel>  
...
```

3. 添加 `initrd` 元素以及主机上 initrd 文件的路径：

```
...  
<initrd>/root/f8-i386-initrd</initrd>  
...
```

4. 启动 VM 以从新内核引导：

```
tux > sudo virsh start sles15
```

15.6 配置内存分配

您还可以使用 `virsh` 来配置分配给 VM Guest 的内存量。此值储存在 `memory` 元素中。请遵循以下步骤：

1. 打开 VM Guest 的 XML 配置：

```
tux > sudo virsh edit sles15
```

2. 搜索 `memory` 元素并设置分配的 RAM 量：

```
...  
<memory unit='KiB'>524288</memory>  
...
```

3. 运行以下命令检查 VM 中分配的 RAM 量：

```
tux > cat /proc/meminfo
```

15.7 添加 PCI 设备

要使用 `virsh` 将 PCI 设备指派到 VM Guest，请执行以下步骤：

1. 标识要指派到 VM Guest 的主机 PCI 设备。在下面的示例中，我们要将一块 DEC 网卡指派到 Guest：

```
tux > sudo lspci -nn
```

```
[...]
03:07.0 Ethernet controller [0200]: Digital Equipment Corporation DECchip \
21140 [FasterNet] [1011:0009] (rev 22)
[...]
```

请记住设备 ID（在本例中为 03:07.0）。

2. 使用 **virsh nodedev-dumpxml ID** 收集有关设备的详细信息。要获取 ID，请将设备 ID (03:07.0) 中的冒号和句点替换为下划线。使用 “pci_0000_” 作为结果的前缀： pci_0000_03_07_0。

```
tux > sudo virsh nodedev-dumpxml pci_0000_03_07_0
<device>
  <name>pci_0000_03_07_0</name>
  <path>/sys/devices/pci0000:00/0000:00:14.4/0000:03:07.0</path>
  <parent>pci_0000_00_14_4</parent>
  <driver>
    <name>tulip</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>3</bus>
    <slot>7</slot>
    <function>0</function>
    <product id='0x0009'>DECchip 21140 [FasterNet]</product>
    <vendor id='0x1011'>Digital Equipment Corporation</vendor>
    <numa node='0' />
  </capability>
</device>
```

记下域、总线和功能的值（请查看上面以粗体列显的 XML 代码）。

3. 从主机系统上分离设备，然后将其挂接到 VM Guest：

```
tux > sudo virsh nodedev-detach pci_0000_03_07_0
Device pci_0000_03_07_0 detached
```



提示：多功能 PCI 设备

使用不支持 FLR（功能级重置）或 PM（电源管理）重置的多功能 PCI 设备时，需要从 VM 主机服务器分离其所有功能。出于安全原因，整个设备都必须重置。如果 VM 主机服务器或其他 VM Guest 仍在使用该设备的某个功能，libvirt 将拒绝指派该设备。

4. 将域、总线、插槽和功能值从十进制转换为十六进制。在本示例中，域 = 0，总线 = 3，插槽 = 7，功能 = 0。确保按正确顺序插入值：

```
tux > printf "<address domain='0x%x' bus='0x%x' slot='0x%x' function='0x%x' />\n" 0 3 7 0
```

这会返回以下结果：

```
<address domain='0x0' bus='0x3' slot='0x7' function='0x0' />
```

5. 在您的域上运行 virsh edit，并使用上一步的结果在 <devices> 部分添加以下设备项：

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0' bus='0x03' slot='0x07' function='0x0' />
  </source>
</hostdev>
```



提示：受管模式与不受管模式的比较

libvirt 可识别 PCI 设备的两种处理模式：受管 或 不受管。在受管模式下，libvirt 将处理从现有驱动程序取消绑定设备（如果需要）、重置设备、在启动域之前将设备绑定到 vfio-pci 等事项的所有细节。对于受管设备，当终止域或者从域中去除设备时，libvirt 会将设备从 vfio-pci 取消绑定，然后将其重新绑定到原始驱动程序。如果设备不受管，则用户必须确保在将设备指派到域之前以及在设备不再由域使用之后，所有这些设备管理方面的操作都已完成。

在上面的示例中，`managed='yes'` 选项表示设备是受管的。要将设备切换为不受管模式，请在上面的列表中设置 `managed='no'`。如果这样做，需使用 `virsh nodedev-detach` 和 `virsh nodedev-reattach` 命令处理相关的驱动程序。在启动 VM Guest 之前，需运行 `virsh nodedev-detach pci_0000_03_07_0` 以从主机分离设备。如果 VM Guest 未运行，您可以运行 `virsh nodedev-reattach pci_0000_03_07_0`，使设备可供主机使用。

6. 关闭 VM Guest，并禁用 SELinux（如果它正在主机上运行）。

```
tux > sudo setsebool -P virt_use_sysfs 1
```

7. 启动 VM Guest 以使指派的 PCI 设备可用：

```
tux > sudo virsh start sles15
```

15.8 添加 USB 设备

要使用 `virsh` 将 USB 设备指派到 VM Guest，请执行以下步骤：

1. 标识要指派到 VM Guest 的主机 USB 设备：

```
tux > sudo lsusb
[...]
Bus 001 Device 003: ID 0557:2221 ATEN International Co., Ltd Winbond Hermon
[...]
```

记下供应商 ID 和产品 ID。在本示例中，供应商 ID 为 `0557`，产品 ID 为 `2221`。

2. 在您的域上运行 `virsh edit`，并使用上一步获得的值在 `<devices>` 部分添加以下设备项：

```
<hostdev mode='subsystem' type='usb'>
  <source startupPolicy='optional'>
    <vendor id='0557' />
    <product id='2221' />
  </source>
```



```
</hostdev>
```



提示：供应商/产品或设备地址

如果不使用 `<供应商/>` 和 `<产品/>` ID 来定义主机设备，可以使用 `<address/>` 元素，具体请参见第 15.7 节“添加 PCI 设备”中的主机 PCI 设备相关内容。

3. 关闭 VM Guest，并禁用 SELinux（如果它正在主机上运行）：

```
tux > sudo setsebool -P virt_use_sysfs 1
```

4. 启动 VM Guest 以使指派的 PCI 设备可用：

```
tux > sudo virsh start sles15
```

15.9 添加 SR-IOV 设备

支持单根 I/O 虚拟化 (SR-IOV) 的 PCIe 设备能够复制其资源，因此它们看上去像是多个设备。可将其中的每个“伪设备”指派到 VM Guest。

SR-IOV 是外围部件互连专业组 (PCI-SIG) 联盟制定的行业规范。其中介绍了物理功能 (PF) 和虚拟功能 (VF)。PF 是用于管理和配置设备的完整 PCIe 功能。PF 还可以移动数据。VF 在配置和管理方面的作用有所欠缺 — 它们只能移动数据，提供的配置功能有限。由于 VF 不包括所有的 PCIe 功能，主机操作系统或超级管理程序必须支持 SR-IOV 才能访问和初始化 VF。理论上 VF 的最大数量为每台设备 256 个（因此，对于双端口以太网卡，最大数量为 512 个）。在实际环境中，此最大数量要少得多，因为每个 VF 都会消耗资源。

15.9.1 要求

要使用 SR-IOV，必须符合以下要求：

- 支持 SR-IOV 的网卡（从 SUSE Linux Enterprise Server 15 开始，只有网卡支持 SR-IOV）
- 支持硬件虚拟化的 AMD64/Intel 64 主机（AMD-V 或 Intel VT-x），有关详细信息，请参见第 7.3 节“KVM 硬件要求”

- 支持设备指派的芯片组（AMD-Vi 或 Intel VT-d）
- [libvirt 0.9.10](#) 或更高版本
- 主机系统上必须装载并配置 [SR-IOV](#) 驱动程序
- 符合[重要：VFIO 和 SR-IOV 的要求](#)中所列要求的主机配置
- 要指派到 VM Guest 的 VF 的 PCI 地址列表



提示：检查设备是否支持 SR-IOV

可以通过运行 **lspci** 从设备的 PCI 描述符中获取有关该设备是否支持 SR-IOV 的信息。支持 [SR-IOV](#) 的设备会报告类似如下的功能：

```
Capabilities: [160 v1] Single Root I/O Virtualization (SR-IOV)
```



注意：在创建 VM Guest 时添加 SR-IOV 设备

您必须已按照[第 15.9.2 节“装载和配置 SR-IOV 主机驱动程序”](#)中所述配置 VM 主机服务器，才可在最初设置 VM Guest 时向其添加 SR-IOV 设备。

15.9.2 装载和配置 SR-IOV 主机驱动程序

要访问和初始化 VF，需在主机系统上装载一个支持 SR-IOV 的驱动程序。

1. 在装载驱动程序之前，请运行 **lspci** 来确保可正常检测到网卡。下面的示例显示了双端口 Intel 82576NS 网卡的 **lspci** 输出：

```
tux > 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)
```

如果未检测到网卡，可能是因为未在 BIOS/EFI 中启用硬件虚拟化支持。要检查是否已启用硬件虚拟化支持，请查看主机 BIOS 中的设置。

2. 运行 **lsmod** 来检查是否已装载 **SR-IOV** 驱动程序。在下面的示例中，用于检查是否装载了 Intel 82576NS 网卡的 **igb** 驱动程序的命令返回了一条结果。这表示已装载该驱动程序。如果该命令未返回任何结果，则表示未装载该驱动程序。

```
tux > sudo /sbin/lsmod | egrep "^igb "  
igb                  185649  0
```

3. 如果已装载驱动程序，请跳过以下步骤。如果尚未装载 **SR-IOV** 驱动程序，需要先去除非 **SR-IOV** 驱动程序，然后再装载新驱动程序。使用 **rmmod** 卸载驱动程序。下面的示例会卸载 Intel 82576NS 网卡的非 **SR-IOV** 驱动程序：

```
tux > sudo /sbin/rmmod igbvf
```

4. 随后使用 **modprobe** 命令装载 **SR-IOV** 驱动程序 — 必须指定 VF 参数 (**max_vfs**)：

```
tux > sudo /sbin/modprobe igb max_vfs=8
```

或者，您也可以通过 SYSFS 装载驱动程序：

1. 通过列出以太网设备确定物理 NIC 的 PCI ID：

```
tux > 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. 要启用 VF，请向 **sriov_numvfs** 参数回送需要装载的 VF 数量：

```
tux > sudo echo 1 > /sys/bus/pci/devices/0000:06:00.1/sriov_numvfs
```

3. 校验是否已装载 VF NIC：

```
tux > 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. 获取可用 VF 的最大数量:

```
tux > 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. 创建 /etc/systemd/system/before.service 文件, 用于在引导时通过 SYSFS 装载 VF:

```
[Unit]
Before=
[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/bash -c "echo 1 > /sys/bus/pci/devices/0000:06:00.1/
sriov_numvfs"
# beware, executable is run directly, not through a shell, check the man
pages
# systemd.service and systemd.unit for full syntax
[Install]
# target in which to start the service
WantedBy=multi-user.target
#WantedBy=graphical.target
```

6. 在启动 VM 之前, 需要创建指向 /etc/init.d/after.local 脚本 (用于分离 NIC) 的另一个服务文件 (after-local.service)。否则 VM 将无法启动:

```
[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. 将此文件复制到 /etc/systemd/system。

```
#!/bin/sh
# ...
virsh nodedev-detach pci_0000_06_08_0
```

将它另存为 /etc/init.d/after.local。

8. 重引导计算机，然后根据本过程的第一步所述，重新运行 **lspci** 命令检查是否已装载 SR-IOV 驱动程序。如果已成功装载 SR-IOV 驱动程序，您应该会看到额外的 VF 行：

```
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)
[...]
```

15.9.3 将 VF 网络设备添加到 VM Guest

在 VM 主机服务器上正确设置 SR-IOV 硬件后，便可将 VF 添加到 VM Guest。为此，您需要先收集一些数据。

过程 15.2：将 VF 网络设备添加到现有 VM GUEST

下面的过程使用的是示例数据。请务必将其替换为您设置中的相应数据。

1. 使用 **virsh nodedev-list** 命令获取您要指派的 VF 的 PCI 地址及其对应的 PF。第 15.9.2 节“装载和配置 SR-IOV 主机驱动程序”中所示的 **lspci** 输出中的数字值（例如 01:00.0 或 04:00.1）已经过转换：添加了前缀“pci_0000_”并将冒号和句点替换为下划线。因此，**lspci** 列出的 PCI ID “04:00.0”会被 virsh 列为“pci_0000_04_00_0”。下面的示例列出了 Intel 82576NS 网卡的第二个端口的 PCI ID：

```
tux > 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
```

前两个项表示 **PF**，其他项表示 **VF**。

2. 对您要添加的 VF 的 PCI ID 运行以下 `virsh nodedev-dumpxml` 命令：

```
tux > 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>
```

下一步需要以下数据：

- <domain>0</domain>
- <bus>4</bus>
- <slot>16</slot>
- <function>0</function>

3. 创建一个临时 XML 文件（例如 `/tmp/vf-interface.xml`），其中包含将 VF 网络设备添加到现有 VM Guest 所需的数据。该文件至少需包含如下所示的内容：

```
<interface type='hostdev'>❶
  <source>
    <address type='pci' domain='0' bus='11' slot='16' function='0'2/>❷
  </source>
</interface>
```

- ❶ VF 的 MAC 地址不固定；每次重引导主机后，MAC 地址都会改变。如果以“传统”方式使用 `hostdev` 添加网络设备，每次重引导主机后都需要重新配置 VM Guest 的网络设备，因为 MAC 地址会改变。为了避免此类问题，`libvirt` 引入了 `hostdev` 值，该值可以在指派设备之前设置网络特定的数据。
 - ❷ 请在此处指定上一步中获取的数据。
4. 如果设备已挂接到主机，则无法将它挂接到 VM Guest。要使该设备可供 Guest 使用，请先将它从主机分离：

```
tux > sudo virsh nodedev-detach pci_0000_04_10_0
```

5. 将 VF 接口添加到现有 VM Guest：

```
tux > sudo virsh attach-device GUEST /tmp/vf-interface.xml --OPTION
```

需将 `GUEST` 替换为 VM Guest 的域名、ID 或 UUID。`--OPTION` 可为下列其中一项：

`--persistent`

此选项始终将设备添加到域的永久性 XML 中。此外，如果域正在运行，则会热插入设备。

`--config`

此选项只影响永久性 XML，即使域正在运行也是如此。设备在下次引导后才会显示在 VM Guest 中。

`--live`

此选项只影响运行中的域。如果域处于非活动状态，则操作将会失败。设备不会永久保留在 XML 中，因此其下次引导后在 VM Guest 中将不可用。

`--current`

此选项影响域的当前状态。如果域处于非活动状态，设备将添加到永久性 XML 中，因此其下次引导后仍可用。如果域处于活动状态，则会热插入设备，但不会将其添加到永久性 XML 中。

6. 要分离 VF 接口，请使用 `virsh detach-device` 命令，该命令也接受上面所列的选项。

15.9.4 动态分配池中的 VF

如果您按照第 15.9.3 节 “将 VF 网络设备添加到 VM Guest” 中所述以静态方式在 VM Guest 的配置中定义了 VF 的 PCI 地址，此类 Guest 将很难迁移到另一台主机。该主机必须在 PCI 总线上的相同位置具有相同的硬件，否则每次启动之前都必须修改 VM Guest 配置。

另一种方法是使用一个包含 SR-IOV 设备所有 VF 的设备池创建 `libvirt` 网络。之后，VM Guest 将引用此网络，每次 VM Guest 启动时，系统都会向它动态分配单个 VF。当 VM Guest 停止时，该 VF 将返回到池中，可供其他 Guest 使用。

15.9.4.1 在 VM 主机服务器上使用 VF 池定义网络

下面的网络定义示例为 SR-IOV 设备创建了一个包含所有 VF 的池，该设备的物理功能 (PF) 位于主机中的网络接口 `eth0` 上：

```
<network>
  <name>passthrough</name>
  <forward mode='hostdev' managed='yes'>
    <pf dev='eth0' />
  </forward>
</network>
```

要在主机上使用此网络，请将上述代码保存到文件（例如 `/tmp/passthrough.xml`）中，然后执行以下命令。请记住将 `eth0` 替换为 SR-IOV 设备的 PF 的实际网络接口名称：

```
tux > sudo virsh net-define /tmp/passthrough.xml
tux > sudo virsh net-autostart passthrough
tux > sudo virsh net-start passthrough
```

15.9.4.2 将 VM Guest 配置为使用池中的 VF

下面的 VM Guest 设备接口定义示例使用第 15.9.4.1 节 “在 VM 主机服务器上使用 VF 池定义网络” 中为 SR-IOV 设备创建的池中的某个 VF。Guest 首次启动时，`libvirt` 会自动派生与该 PF 关联的所有 VF 的列表。

```
<interface type='network'>
  <source network='passthrough'>
```

```
</interface>
```

在第一个使用以 VF 池定义的网络的 VM Guest 启动后，校验关联的 VF 列表。为此，请在主机上运行 **virsh net-dumpxml passthrough**。

```
<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>
```

15.10 列出挂接的设备

尽管 **virsh** 中没有任何机制可列出 VM 主机服务器中已挂接到其 VM Guest 的所有设备，但您可以通过运行以下命令列出已挂接到特定 VM Guest 的所有设备：

```
virsh dumpxml VMGUEST_NAME | xpath -e /domain/devices/hostdev
```

例如：

```
tux > 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>

```



提示：列出通过 `<interface type='hostdev'>` 挂接的 SR-IOV 设备

对于通过 `<interface type='hostdev'>` 挂接到 VM 主机服务器的 SR-IOV 设备，需要使用不同的 XPath 查询：

```
virsh dumpxml VMGUEST_NAME | xpath -e /domain/devices/interface/@type
```

15.11 配置储存设备

储存设备定义于 `disk` 元素中。常规的 `disk` 元素支持多个属性。下面是两个最重要的属性：

- 属性 `type` 描述虚拟磁盘设备的来源。有效值为 `file`、`block`、`dir`、`network` 或 `volume`。
- 属性 `device` 指示如何向 VM Guest 操作系统公开磁盘。例如，可能的值可能包括 `floppy`、`disk`、`cdrom` 及其他。

下面是最重要的子元素：

- `driver` 包含驱动程序和总线。VM Guest 使用驱动程序和总线来操作新磁盘设备。
- 元素 `target` 包含新磁盘显示在 VM Guest 中时所用的设备名称。它还包含可选的总线属性，该属性定义用于操作新磁盘的总线的类型。

下面的过程说明如何将储存设备添加到 VM Guest：

1. 编辑现有 VM Guest 的配置：

```
tux > sudo virsh edit sles15
```

2. 在 `disk` 元素中添加 `disk` 元素以及 `type` 和 `device` 属性：

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

3. 指定 `driver` 元素并使用默认值：

```
<driver name='qemu' type='qcow2' />
```

4. 创建一个磁盘映像，用作新虚拟磁盘设备的来源：

```
tux > sudo qemu-img create -f qcow2 /var/lib/libvirt/images/sles15.qcow2  
32G
```

5. 添加磁盘来源的路径：

```
<source file='/var/lib/libvirt/images/sles15.qcow2' />
```

6. 定义 VM Guest 中的目标设备名以及用于操作磁盘的总线：

```
<target dev='vda' bus='virtio' />
```

7. 重新启动您的 VM：

```
tux > sudo virsh start sles15
```

现在，新储存设备在 VM Guest 操作系统中应该可供使用。

15.12 配置控制器设备

一般情况下，**libvirt** 会根据 VM Guest 使用的虚拟设备类型自动管理控制器。如果 VM Guest 包含 PCI 和 SCSI 设备，libvirt 将自动创建并管理 PCI 和 SCSI 控制器。**libvirt** 还可为特定于超级管理程序的控制器（例如 KVM VM Guest 的 `virtio-serial` 控制器，或 Xen VM

Guest 的 `xenbus` 控制器) 建模。尽管默认控制器及其配置在一般情况下都可满足需求, 但在某些用例中, 需要手动调整控制器或其属性。例如, `virtio-serial` 控制器可能需要更多端口, 或者 `xenbus` 控制器可能需要更多内存或更多虚拟中断。

`Xenbus` 控制器的独特之处在于, 它充当着所有 `Xen` 半虚拟设备的控制器。如果 VM Guest 包含许多磁盘和/或网络设备, 则控制器可能需要更多内存。`Xen` 的 `max_grant_frames` 属性设置要将多少授权帧或共享内存块分配给每个 VM Guest 的 `xenbus` 控制器。

默认值 32 在大多数情况下已够用, 但包含大量 I/O 设备的 VM Guest 以及 I/O 密集型工作负载可能会由于授权帧耗尽而发生性能问题。可以使用 **`xen-diag`** 来检查 `dom0` 与 VM Guest 的当前和最大 `max_grant_frames` 值。VM Guest 必须正在运行:

```
tux > sudo virsh list
 Id   Name           State
-----
 0    Domain-0       running
 3    sle15sp1       running

tux > sudo xen-diag gnttab_query_size 0
domid=0: nr_frames=1, max_nr_frames=256

tux > sudo xen-diag gnttab_query_size 3
domid=3: nr_frames=3, max_nr_frames=32
```

`sle15sp1` Guest 仅使用了 32 个帧中的 3 个。如果您发现了性能问题并且有日志项指出帧数不足, 请使用 **`virsh`** 增大该值。查看 Guest 配置文件中的 `<controller type='xenbus'>` 行, 并添加 `maxGrantFrames` 控制元素:

```
tux > sudo virsh edit sle15sp1
<controller type='xenbus' index='0' maxGrantFrames='40' />
```

保存更改并重新启动 Guest。现在, `xen-diag` 命令应该会显示您的更改:

```
tux > sudo xen-diag gnttab_query_size 3
domid=3: nr_frames=3, max_nr_frames=40
```

与 maxGrantFrames 类似，xenbus 控制器也支持 `maxEventChannels`。事件通道类似于半虚拟中断，它们与授权帧共同构成半虚拟驱动程序的数据传输机制。它们还用于处理器间的中断。包含大量 vCPU 和/或许多半虚拟设备的 VM Guest 可能需要增大最大默认值 1023。可以像更改 maxGrantFrames 那样更改 maxEventChannels：

```
tux > sudo virsh edit sle15sp1
<controller type='xenbus' index='0' maxGrantFrames='128'
maxEventChannels='2047' />
```

有关详细信息，请参见 <https://libvirt.org/formatdomain.html#elementsControllers> 上 libvirt 的《Domain XML format》（域 XML 格式）手册中的“Controllers”（控制器）一节。

15.13 配置视频设备

使用虚拟机管理器时，只能定义视频设备型号。只能在 XML 配置中更改分配的 VRAM 量或 2D/3D 加速。

15.13.1 更改分配的 VRAM 量

1. 编辑现有 VM Guest 的配置：

```
tux > sudo virsh edit sles15
```

2. 更改分配的 VRAM 大小：

```
<video>
<model type='vga' vram='65535' heads='1'>
...
</model>
</video>
```

3. 通过查看虚拟机管理器中的数量来检查 VM 中的 VRAM 量是否已更改。

15.13.2 更改 2D/3D 加速状态

1. 编辑现有 VM Guest 的配置：

```
tux > sudo virsh edit sles15
```

2. 要启用/禁用 2D/3D 加速，请相应地更改 `accel3d` 和 `accel2d` 的值：

```
<video>
<acceleration accel3d='yes' accel2d='no'>
...
</model>
</video>
```



提示：启用 2D/3D 加速

只有 `vbox` 视频设备支持 2D/3D 加速。无法在其他视频设备上启用此功能。

15.14 配置网络设备

本节介绍如何使用 `virsh` 配置虚拟网络设备的特定方面。

<https://libvirt.org/formatdomain.html#elementsDriverBackendOptions> 中提供了有关 `libvirt` 网络接口规范的更多细节。

15.14.1 使用多队列 virtio-net 提升网络性能

多队列 virtio-net 功能允许 VM Guest 的虚拟 CPU 并行传输包，因此可以提升网络性能。有关更多一般信息，请参见第 30.3.3 节“使用多队列 virtio-net 提升网络性能”。

要为特定的 VM Guest 启用多队列 virtio-net，请遵照第 15.1 节“编辑 VM 配置”中所述编辑其 XML 配置，并按如下所示修改其网络接口：

```
<interface type='network'>
[...]
```

```
<model type='virtio' />
<driver name='vhost' queues='NUMBER_OF_QUEUES' />
</interface>
```

15.15 使用 Macvtap 共享 VM 主机服务器网络接口

使用 Macvtap 可将 VM Guest 虚拟接口直接挂接到主机网络接口。基于 macvtap 的接口扩展了 VM 主机服务器网络接口，它在相同以太网段上有自己的 MAC 地址。通常，使用此功能是为了使 VM Guest 和 VM 主机服务器都直接显示在 VM 主机服务器连接的交换机上。



注意：Macvtap 不能与 Linux 网桥搭配使用

Macvtap 不能与已连接到 Linux 网桥的网络接口搭配使用。在尝试创建 macvtap 接口之前，请去除网桥中的接口。



注意：使用 Macvtap 实现 VM Guest 到 VM 主机服务器的通讯

使用 macvtap 时，一个 VM Guest 可与其他多个 VM Guest 通讯，并可与网络上的其他外部主机通讯。但是，该 VM Guest 无法与用于运行它的 VM 主机服务器通讯。这是规定的 macvtap 行为，原因与 VM 主机服务器物理以太网挂接到 macvtap 网桥的方式有关。从 VM Guest 进入该网桥并转发到物理接口的流量无法回弹到 VM 主机服务器的 IP 堆栈。同样，来自 VM 主机服务器 IP 堆栈并发送到物理接口的流量无法回弹到 macvtap 网桥以转发到 VM Guest。

libvirt 通过指定接口类型 `direct` 支持基于 macvtap 的虚拟网络接口。例如：

```
<interface type='direct'>
  <mac address='aa:bb:cc:dd:ee:ff' />
  <source dev='eth0' mode='bridge' />
  <model type='virtio' />
</interface>
```

可以使用 `mode` 属性控制 macvtap 设备的操作模式。下面的列表显示了该属性的可能的值以及每个值的说明：

- vepa：将所有 VM Guest 包都发送到外部网桥。如果包的目标是某个 VM Guest，而该 VM Guest 所在的 VM 主机服务器与包的来源服务器相同，那么这些包将由支持 VEPA 的网桥（现今的网桥通常都不支持 VEPA）发回到该 VM 主机服务器。
- bridge：将其目标与来源为同一 VM 主机服务器的包直接递送到目标 macvtap 设备。来源和目标设备需处于 bridge 模式才能直接递送。如果其中一个设备处于 vepa 模式，则需要使用支持 VEPA 的网桥。
- private：将所有包都发送到外部网桥；仅当通过外部路由器或网关发送所有包，并且设备会将其发回到 VM 主机服务器时，才将所有包递送到同一 VM 主机服务器上的目标 VM Guest。如果来源或目标设备处于 private 模式，将遵循此过程。
- passthrough：可为网络接口提供更强大的能力的一种特殊模式。将所有包转发到接口，并允许 virtio VM Guest 更改 MAC 地址或设置混杂模式，以桥接该接口或在该接口上创建 VLAN 接口。请注意，在 passthrough 模式下，网络接口不可共享。将某个接口指派到 VM Guest 会使其与 VM 主机服务器断开连接。出于此原因，在 passthrough 模式下常常会将 SR-IOV 虚拟功能指派到 VM Guest。

15.16 禁用内存气球设备

内存气球已成为 KVM 的默认选项。设备将显式添加到 VM Guest，因此您无需在 VM Guest 的 XML 配置中添加此元素。但是，如果您出于任何原因想要在 VM Guest 中禁用内存气球，需按如下所示设置 model='none'：

```
<devices>
  <memballoon model='none' />
</device>
```

15.17 配置多个监视器（双头）

libvirt 支持使用双头配置在多个监视器上显示 VM Guest 的视频输出。

！ 重要：不受 Xen 支持

Xen 超级管理程序不支持双头配置。

过程 15.3：配置双头

1. 当虚拟机正在运行时，请校验 `xf86-video-qxl` 软件包是否已安装在 VM Guest 中：

```
tux > rpm -q xf86-video-qxl
```

2. 关闭 VM Guest，并按照第 15.1 节“编辑 VM 配置”中所述开始编辑其 XML 配置。
3. 校验虚拟显卡的型号是否为“qxl”：

```
<video>
  <model type='qxl' ... />
```

4. 将显卡型号规格中的 `heads` 参数从默认值 `1` 增大为 `2`，例如：

```
<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. 将虚拟机配置为使用 Spice 显示器而不是 VNC：

```
<graphics type='spice' port='5916' autoport='yes' listen='0.0.0.0'>
  <listen type='address' address='0.0.0.0' />
</graphics>
```

6. 启动虚拟机并使用 `virt-viewer` 连接到其显示器，例如：

```
tux > virt-viewer --connect qemu+ssh://USER@VM_HOST/system
```

7. 在 VM 列表中，选择您已修改了其配置的 VM，并单击连接确认。

8. 在 VM Guest 中装载图形子系统 (Xorg) 后，选择视图 > 显示器 > 显示器 2 打开一个新窗口，其中会显示第二个监视器的输出。

16 使用 Vagrant 管理虚拟机

Vagrant 工具为创建、部署和管理虚拟开发环境提供统一的工作流程。下列章节介绍如何使用 Vagrant 管理虚拟机。

16.1 Vagrant 简介

Vagrant 通过一个简单的配置文件为各种虚拟化提供程序提供了一个抽象层，可让开发人员和操作人员快速启动运行 Linux 或任何其他操作系统的虚拟机 (VM)。

16.1.1 Vagrant 概念

Vagrant 使用 提供器、配置器、盒子 (box) 及 Vagrantfile 作为虚拟机的组建模块。

VAGRANT 术语

提供器

用于设置和创建虚拟环境的服务。Vagrant 原生支持 VirtualBox 和 Microsoft Hyper-V 虚拟化以及 Docker 容器。通过插件来支持 libvirt、VMware 或 AWS 等其他服务。


配置器

用于自定义虚拟环境配置的工具。Vagrant 不仅构建了内置的提供器用于上载文件、同步目录或执行外壳命令，同时还支持 Ansible、CFEngine、Chef、Puppet 和 Salt 等配置管理系统。

Vagrantfile

虚拟环境的配置文件和文件名 (Vagrantfile)。其中包含计算机和软件要求，以及创建可直接用于开发的盒子所需执行的所有步骤。

盒子

虚拟环境的格式和扩展名 (*.box)。可以从 Vagrant Cloud (<https://vagrantcloud.com/>)  下载盒子，也可将盒子从一台计算机复制到另一台计算机，以复制环境。

SUSE 使用 VirtualBox 和 `libvirt` 提供器提供适用于 SUSE Linux Enterprise 的官方 Vagrant 盒子。SUSE Linux Enterprise Server 盒子适用于 AMD64/Intel 64 和 AArch64 体系结构，而 SUSE Linux Enterprise Desktop 仅适用于 AMD64/Intel 64。

16.1.2 Vagrant 示例

可以使用 Vagrant 通过下面一组命令启动新 VM。此示例使用适用于 openSUSE Tumbleweed 的官方 Vagrant 盒子（可从 [Vagrant Cloud \(https://vagrantcloud.com/\)](https://vagrantcloud.com/) 获取）。

过程 16.1：使用 OPENSUSE TUMBLEWEED 创建 VAGRANT 环境

1. 下载适用于 openSUSE Tumbleweed 的 Vagrant 盒子：

```
vagrant init opensuse/Tumbleweed.x86_64
```

这还会在 Vagrant 中注册该盒子并创建 `Vagrantfile`。

2. （可选）编辑 `Vagrantfile` 以自定义环境。

3. 启动盒子：

```
vagrant up
```

4. 通过 `ssh` 访问盒子：

```
vagrant ssh
```

16.2 适用于 SUSE Linux Enterprise 的 Vagrant 盒子

从 SUSE Linux Enterprise 15 SP2 开始，SUSE 使用 VirtualBox 和 `libvirt` 提供器提供适用于 SUSE Linux Enterprise 的官方 Vagrant 盒子。SUSE Linux Enterprise Server 盒子适用于 AMD64/Intel 64 和 AArch64 体系结构，而 SUSE Linux Enterprise Desktop 仅适用于 AMD64/Intel 64。

为缩减大小，这些盒子打包在极其精简的软件包中，它们并未注册，因此用户在进一步配置之前需要注册盒子。

只能从 <https://download.suse.com> 直接下载盒子。因此，必须按如下所示将下载的盒子手动注册到 Vagrant 中：

```
vagrant box add --name SLES-15-SP2 \  
/path/to/SLES15-SP2-Vagrant.x86_64-15.2-libvirt-*.vagrant.libvirt.box
```

然后，该盒子的名称将会显示为 SLES-15-SP2，您可以像使用其他 Vagrant 盒子一样使用它：

```
vagrant init SLES-15-SP2  
vagrant up  
vagrant ssh
```

16.3 更多资料

有关 Vagrant 及其配置的详细信息，请参见 <https://docs.vagrantup.com/> 上的官方文档。

III 独立于超级管理程序的功能

- 17 磁盘缓存模式 176
- 18 VM Guest 时钟设置 180
- 19 libguestfs 182
- 20 QEMU Guest 代理 195

17 磁盘缓存模式

17.1 磁盘接口缓存模式

超级管理程序允许在配置 VM Guest 时指定各种储存缓存策略。可为每个 Guest 磁盘接口指定以下缓存模式之一：`writethrough`、`writeback`、`none`、`directsync` 或 `unsafe`。如果未指定任何模式，将使用相应的默认缓存模式。这些缓存模式会影响基于主机的储存的访问方式，如下所述：

- 读取/写入数据可以缓存在主机页缓存中。
- Guest 的储存控制器会获悉是否存在写入缓存，以便能够使用刷新命令。
- 可以使用同步写入模式，在此模式下，仅当已将写入请求提交到储存设备时，才将这些请求报告为完成。
- 可出于性能原因忽略刷新命令（由 Guest 储存控制器生成）。

如果 Guest 与其储存设备之间的连接无序断开，则使用的缓存模式将影响到是否会发生数据丢失。缓存模式还会显著影响磁盘性能。此外，某些缓存模式与实时迁移不兼容，具体取决于多个因素。有关缓存模式、磁盘映像格式、映像位置和储存子系统的哪种组合是最佳的，并没有简单的规则可循。用户应该精心规划每个 Guest 的配置，并试验各种配置来确定最佳性能。

17.2 缓存模式的说明

未指定缓存模式

在较早的 QEMU 版本中，不指定缓存模式意味着使用 `writethrough` 作为默认模式。在 SUSE Linux Enterprise Server 随附的新版本中，已将各种 Guest 储存接口固定下来，可以更正确地处理 `writeback` 或 `writethrough` 语义。这样就可以将默认缓存模式切换到 `writeback`。`ide`、`scsi` 和 `virtio` 的 Guest 驱动程序必须各尽所能地禁用写回缓存，以使所用的缓存模式还原为 `writethrough`。不过，典型的 Guest 储存驱动程序会将默认缓存模式保留为 `writeback`。

writethrough (直写)

此模式会导致超级管理程序使用 `O_DSYNC` 语义来与磁盘映像文件或块设备交互。仅当已将数据提交到储存设备后，才将写入操作报告为已完成。以称为直写的缓存模式来使用主机页缓存。Guest 的虚拟储存适配器将获悉不存在写回缓存，因此 Guest 无需发送刷新命令来管理数据完整性。储存设备会像存在直写缓存一样工作。

writeback (写回)

此模式会导致超级管理程序既不使用 `O_DSYNC`，也不使用 `O_DIRECT` 语义来与磁盘映像文件或块设备交互。将使用主机页缓存，在将写入内容放入主机页缓存后，即会向 Guest 报告写入操作已完成。常规页缓存管理仍会处理向储存设备提交内容的操作。此外，会向 Guest 的虚拟储存适配器告知写回缓存，因此 Guest 预期会按需发送刷新命令来管理数据完整性。类似于具有 RAM 缓存的 raid 控制器。

none (无)

此模式会导致超级管理程序使用 `O_DIRECT` 语义来与磁盘映像文件或块设备交互。将绕过主机页缓存，I/O 直接在超级管理程序用户空间缓冲区与储存设备之间发生。由于仅当已将写入内容放入其写入队列后，实际储存设备才能将写入操作报告为已完成，因此 Guest 的虚拟储存适配器将获悉存在写回缓存这一情况。Guest 预期会按需发送刷新命令来管理数据完整性。以性能为导向，相当于直接访问主机的磁盘。

unsafe (不安全)

此模式与前面所述的 `writeback` 模式类似。此“unsafe”模式的主要特定是，将忽略来自 Guest 的所有刷新命令。使用此模式意味着用户同意性能优先，可以接受主机发生故障时数据可能丢失的风险。在安装 Guest 等情形下很有用，但对于生产工作负载并不适合。

directsync (直接同步)

此模式会导致超级管理程序使用 `O_DSYNC` 和 `O_DIRECT` 语义来与磁盘映像文件或块设备交互。这意味着，仅当已将数据提交到储存设备，同时还需要绕过主机页缓存时，才将写入操作报告为已完成。与 `writethrough` (直写) 一样，此模式对于不按需发送刷新命令的 Guest 很有用。它是新增的最后一缓存模式，补全了缓存和直接访问语法的可能组合。

17.3 缓存模式对数据完整性的影响

writethrough、none、directsync

这些是最安全的模式。如果 Guest 操作系统是“新式版本且行为正常”（即，可按需使用刷新命令），则认为这三种模式是同等安全的。如果您怀疑 Guest 的安全性，请使用 writethrough 或 directsync。请注意，某些文件系统与 none 或 directsync 不兼容，因为它们不支持这些缓存模式所依赖的 O_DIRECT。

writeback

此模式告知 Guest 存在写入缓存，并依赖于 Guest 按需发送刷新命令来维持其磁盘映像中的数据完整性。这是新式文件系统中充分纳入考虑的一种常见储存设计。此模式会使 Guest 在主机发生故障时（不太可能发生）面临数据丢失的风险，因为将写入操作报告为已完成与将写入内容提交到储存设备之间存在时间差。

unsafe

此模式与 writeback 缓存类似，只不过存在以下差别：将忽略 Guest 刷新命令，因而消除了这些刷新命令的数据完整性控制作用，导致因主机故障而造成的数据丢失风险增大。应该将“unsafe”（不安全）这一名称视为一种警告，与其他模式相比，使用此模式在主机发生故障时丢失数据的概率要高得多。Guest 终止时，缓存的数据会随即刷新。

17.4 缓存模式对性能的影响

充分利用页缓存、直写页缓存或者完全绕过页缓存的选项可能会对性能产生重大影响。其他影响磁盘性能的因素包括实际的储存系统的功能、使用的磁盘映像格式、页缓存的潜在大小，以及使用的 IO 调度程序。此外，不刷新写入缓存可提高性能，但也会带来上面所述的风险。一般而言，高端系统在使用缓存模式 none 时通常可以获得最佳性能，因为此模式减少了需进行的数据复制操作。此外，还应该考虑到让多个 Guest 共享通用主机页缓存可能带来的优势，读写比例以及 AIO 模式 native 的使用（参见下文）。

17.5 缓存模式对实时迁移的影响

缓存储存数据和元数据会限制支持实时迁移的配置。目前，只能使用 raw 和 qcow2 映像格式进行实时迁移。如果使用群集文件系统，则所有缓存模式都支持实时迁移。否则，只有 none 缓存模式支持在读取/写入共享储存中进行实时迁移。

libvirt 管理层包含根据多种因素检查迁移兼容性的功能。如果 Guest 储存托管在群集文件系统上，并且是只读的或者标记为可共享，则在确定是否允许迁移时会忽略缓存模式。除非将缓存模式设置为 none，否则 libvirt 不允许迁移。不过，您可以通过在迁移 API 中使用 “unsafe” 选项来覆盖此限制，此选项也受 virsh 的支持，如下面的示例所示

```
tux > virsh migrate --live --unsafe
```



提示

要设置 AIO 模式 native，缓存模式需设为 none。如果使用另一种缓存模式，则 AIO 模式将无提示切换回默认值 threads。主机中的 Guest 刷新通过 fdatasync() 来实现。

18 VM Guest 时钟设置

在 VM Guest 中保持准确的时间是虚拟化的一项较为困难的工作。保持准确的时间对于网络应用程序特别重要，也是进行 VM Guest 实时迁移的先决条件。



提示：VM 主机服务器上的计时

强烈建议在 VM 主机服务器上也保持准确的时间，例如，通过使用 NTP 来实现（有关详细信息，请参见《管理指南》，第 30 章“使用 NTP 同步时间”）。

18.1 KVM：使用 `kvm_clock`

KVM 提供通过 `kvm_clock` 驱动程序支持的半虚拟化时钟。强烈建议使用 `kvm_clock`。

在运行 Linux 的 VM Guest 中使用以下命令来检查是否已装载 `kvm_clock` 驱动程序：

```
tux > 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
```

要检查当前使用了哪个时钟源，请在 VM Guest 中运行以下命令。此命令应该会输出 `kvm-clock`：

```
tux > cat /sys/devices/system/clocksource/clocksource0/current_clocksource
```



重要：kvm-clock 和 NTP

使用 `kvm-clock` 时，建议同时在 VM Guest 中使用 NTP，并在 VM 主机服务器上也使用 NTP。

18.1.1 其他计时方法

半虚拟化 `kvm-clock` 目前不适用于 Windows* 操作系统。对于 Windows*，请使用 [Windows 时间服务工具](http://technet.microsoft.com/en-us/library/cc773263%28WS.10%29.aspx) 进行时间同步（有关详细信息，请参见 <http://technet.microsoft.com/en-us/library/cc773263%28WS.10%29.aspx>）。

18.2 Xen 虚拟机时钟设置

在 Xen 4 中，已去除用于在 Xen 主机与 Guest 之间进行时间同步的独立时钟设置 `/proc/sys/xen/independent_wallclock`。引入了新配置选项 `tsc_mode`。此选项指定利用时戳计数器将 Guest 时间与 Xen 服务器同步的方法。其默认值“0”适合绝大多数硬件和软件环境。有关 `tsc_mode` 的更多细节，请参见 `xen-tscmode` 手册页 (**man 7 xen-tscmode**)。

19 libguestfs

虚拟机由磁盘映像和定义文件构成。您可以手动访问和操作这些 Guest 组件（在常规超级管理程序进程外部），但这从本质上会给数据完整性造成危害和风险。libguestfs 是用于安全访问和修改**虚拟机**磁盘映像的一个 C 库和一组相应工具 — 这些操作在常规超级管理程序进程外部进行，却不会像手动编辑那样往往造成风险。




19.1 VM Guest 操作概述

19.1.1 VM Guest 操作风险

磁盘映像和定义文件不过是 Linux 环境中另一种类型的文件，因此可以使用许多工具来访问、编辑这些文件以及向其中写入数据。如果正确使用，此类工具可成为 Guest 管理的重要组成部分。但是，即使是正确使用这些工具，也不一定能够杜绝风险。手动操作 Guest 磁盘映像时应考虑到如下风险：

- **数据损坏**：如果绕过虚拟化保护层，通过主机计算机或群集中的另一节点并发访问映像，可能会导致更改丢失或数据损坏。
- **安全性**：将磁盘映像装入为循环设备需要 root 访问权限。如果映像不是循环装入的，其他用户和进程就有可能可以访问磁盘内容。
- **管理员错误**：正确绕过虚拟化层需要对虚拟组件和工具有深入的了解。如果在做出更改后无法隔离映像或无法正确进行清理，可能会导致在恢复虚拟化控制后出现其他问题。

19.1.2 libguestfs 的设计用途

libguestfs C 库用于安全地创建、访问和修改虚拟机 (VM Guest) 磁盘映像。它还提供适用于 Perl (<http://libguestfs.org/guestfs-perl.3.html>) 、Python (<http://libguestfs.org/guestfs-python.3.html>) 、PHP（仅限 64 位计算机）和 Ruby (<http://libguestfs.org/guestfs-ruby.3.html>)  的其他语言绑定。libguestfs 无需 root 权限即可访问 VM Guest 磁盘映像，并提供多层防御机制来防范恶意磁盘映像。

libguestfs 提供许多用于访问和修改 VM Guest 磁盘映像与内容的工具。这些工具提供如下功能：查看和编辑 Guest 内部的文件、通过脚本对 VM Guest 进行更改、监视已用/可用磁盘空间统计信息、创建 Guest、执行 V2V 或 P2V 迁移、执行备份、克隆 VM Guest、格式化磁盘和调整磁盘大小。



警告：最佳实践

切勿在实时虚拟机上使用 libguestfs 工具，这可能会导致 VM Guest 中发生磁盘损坏。libguestfs 工具会尝试阻止您这样做，但无法做到万无一失。

不过，大多数命令都具有 `--ro`（只读）选项。您可以使用此选项将命令关联到实时虚拟机。结果有时可能比较奇怪或不一致，但可以避免磁盘损坏的风险。

19.2 软件包安装

libguestfs 通过 4 个软件包提供：

- libguestfs0：提供主 C 库
- guestfs-data：包含启动映像时使用的设备文件（储存在 /usr/lib64/guestfs 中）
- guestfs-tools：核心 guestfs 工具、手册页和 /etc/libguestfs-tools.conf 配置文件。
- guestfs-winsupport：在 guestfs 工具中提供对 Windows 文件 Guest 的支持。仅当您需要处理 Windows Guest 时才需安装此软件包，例如，在将 Windows Guest 转换为 KVM 时。

要在系统上安装 guestfs 工具，请运行：

```
tux > sudo zypper in guestfs-tools
```

19.3 Guestfs 工具

19.3.1 修改虚拟机

guestfs-tools 软件包中的工具集用于访问和修改虚拟机磁盘映像。此功能通过用户所熟悉的、可提供内置保护措施来确保映像完整性的外壳界面提供。Guestfs 工具外壳会公开 guestfs API 的所有功能，并使用计算机上安装的软件包以及 `/usr/lib64/guestfs` 中的文件即时创建设备。

19.3.2 支持的文件系统和磁盘映像

Guestfs 工具支持各种文件系统，包括：

- Ext2、Ext3、Ext4
- Xfs
- Btrfs

还支持多种磁盘映像格式：

- raw
- qcow2



警告：不支持的文件系统

Guestfs 可能还支持 Windows* 文件系统（VFAT、NTFS）、BSD* 和 Apple* 文件系统，以及其他磁盘映像格式（VMDK、VHDX...）。但这些文件系统和磁盘映像格式在 SUSE Linux Enterprise Server 上不受支持。

19.3.3 virt-rescue

virt-rescue 类似于救援 CD，但用于虚拟机，且无需提供 CD。virt-rescue 为用户提供救援外壳和一些简单的恢复工具，可用于检查和更正虚拟机或磁盘映像中的问题。


```
tux > 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-do-chroot.sh /dev/sda2

><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):~ #
```

您现在是以救援模式运行 VM Guest 的：

```
(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
```

19.3.4 virt-resize

virt-resize 用于调整虚拟机磁盘的大小（增大或减小其总体大小），以及调整其中包含的任何分区的大小，或删除这些分区。

完整的分步操作示例：如何扩展虚拟机磁盘

1. 首先，在关闭虚拟机的情况下，确定此虚拟机上可用的分区大小：

```
tux > 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** 无法执行就地磁盘修改 — 必须有足够的空间用于储存调整大小后的输出磁盘。使用 **truncate** 命令创建适宜大小的文件：

```
tux > truncate -s 32G outdisk.img
```

- ### 3. 使用 `virt-resize` 调整磁盘映像的大小。 `virt-resize` 要求为输入映像和输出映像指定两个必需的参数：

```
tux > 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%  
[██████████████████████████████████████████████████████████████]=====  
00:03  
  
Expanding /dev/sda1 using the 'resize2fs' method ...  
  
Resize operation completed with no errors. Before deleting the old  
disk, carefully check that the resized disk boots and works correctly.
```

- #### 4. 确认是否已正确调整映像大小:

```
tux > virt-filesystems --long --parts --blkdevs -h -a outdisk.img
```

Name	Type	MBR	Size	Parent
/dev/sda1	partition	83	32G	/dev/sda
/dev/sda	device	-	32G	-

5. 使用新磁盘映像启动 VM Guest，确认其运行正常，然后删除旧映像。

19.3.5 其他 virt-* 工具

某些 guestfs 工具可以简化管理任务 — 例如查看和编辑文件，或者获取有关虚拟机的信息。

19.3.5.1 virt-filesystems

此工具用于报告有关磁盘映像或虚拟机中的文件系统、分区和逻辑卷的信息。

```
tux > virt-filesystems -l -a sles.qcow2
```

Name	Type	VFS	Label	Size	Parent
/dev/sda1	filesystem	ext3	-	17178820608	-

19.3.5.2 virt-ls

virt-ls 可列出虚拟机或磁盘映像中的文件名、文件大小、校验和、扩展属性等信息。可以指定多个目录名，在这种情况下，每个目录的输出将会串联起来。要列出某个 libvirt Guest 中的目录，请使用 `-d` 选项指定 Guest 名称。对于磁盘映像，请使用 `-a` 选项。

```
tux > 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
[...]
```

19.3.5.3 **virt-cat**

virt-cat 命令行工具用于显示命名虚拟机（或磁盘映像）中存在的文件的内容。可以指定多个文件名，在这种情况下，这些文件名会串联到一起。每个文件名都必须是以根目录（“/”）开头的完整路径。

```
tux > virt-cat -a sles.qcow2 /etc/fstab
devpts /dev/pts devpts mode=0620,gid=5 0 0
proc   /proc     proc   defaults      0 0
```

19.3.5.4 **virt-df**

virt-df 命令行工具用于显示虚拟机文件系统上的可用空间。与其他工具不同，它不仅会显示分配给虚拟机的磁盘大小，而且还会查看磁盘映像内部以显示实际使用的空间量。

```
tux > virt-df -a sles.qcow2
Filesystem                                1K-blocks      Used  Available  Use%
sles.qcow2:/dev/sda1                      16381864      520564   15022492    4%
```

19.3.5.5 **virt-edit**

virt-edit 命令行工具能够编辑驻留在命名虚拟机（或磁盘映像）中的文件。

19.3.5.6 **virt-tar-in/out**

virt-tar-in 可将未压缩的 TAR 存档解压缩到虚拟机磁盘映像或命名的 libvirt 域中。**virt-tar-out** 可将虚拟机磁盘映像目录打包成 TAR 存档。

```
tux > virt-tar-out -a sles.qcow2 /home homes.tar
```

19.3.5.7 **virt-copy-in/out**

virt-copy-in 可将本地磁盘中的文件和目录复制到虚拟机磁盘映像或命名的 libvirt 域中。**virt-copy-out** 可从虚拟机磁盘映像或命名的 libvirt 域中复制文件和目录。

```
tux > virt-copy-in -a sles.qcow2 data.tar /tmp/
virt-ls -a sles.qcow2 /tmp/
.ICE-unix
.X11-unix
data.tar
```

19.3.5.8 **virt-log**

virt-log 可显示命名的 libvirt 域、虚拟机或磁盘映像的日志文件。如果安装了 guestfs-winsupport 软件包，virt-log 还可显示 Windows 虚拟机磁盘映像的事件日志。

```
tux > 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>
```

...

19.3.6 **guestfish**

guestfish 是用于检查和修改虚拟机文件系统的外壳和命令行工具。它使用 libguestfs 并公开 guestfs API 的所有功能。

用法示例：

```
tux > 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
```

19.3.7 将物理机转换为 KVM Guest

Libguestfs 提供了可帮助您将 Xen 虚拟机或物理机转换为 KVM Guest 的工具。《Xen 到 KVM 的迁移指南》文章中介绍了从 Xen 转换到 KVM 的方案。下一节将介绍一个特殊用例：将裸机转换为 KVM 计算机。

SUSE Linux Enterprise Server 尚不支持将物理机转换为 KVM 计算机。此功能仅发布为技术预览版。

转换某个物理机需要收集有关该物理机的信息，并将这些信息传输到转换服务器。要实现此目的，需在计算机上运行一个使用 **virt-p2v** 和 **kiwi** 工具准备的实时系统。

过程 19.2：使用 VIRT-P2V

1. 使用以下命令安装所需的软件包：

```
tux > sudo zypper in virt-p2v kiwi-desc-isoboot
```



注意

这些步骤将会阐述如何创建一个用于创建可引导 DVD 的 ISO 映像。或者，您也可以改为创建 PXE 引导映像；有关使用 KIWI 构建 PXE 映像的详细信息，请参见 **man virt-p2v-make-kiwi**。

2. 创建 KIWI 配置：

```
tux > virt-p2v-make-kiwi -o /tmp/p2v.kiwi
```

-o 定义在何处创建 KIWI 配置。

3. 如果需要，请编辑生成的配置中的 **config.xml** 文件。例如，在 **config.xml** 中调整实时系统的键盘布局。
4. 使用 **kiwi** 构建 ISO 映像：

```
tux > kiwi --build /tmp/p2v.kiwi ❶ \  
    -d /tmp/build ❷ \  
    --ignore-repos \  
    --add-repo http://URL/T0/SLE/REPOSITORIES ❸ \  
    --iso
```

```
--type iso
```

- ❶ 存放上一步中生成的 KIWI 配置的目录。
- ❷ KIWI 将用于存放生成的 ISO 映像及其他中间构建结果的目录。
- ❸ 使用 `zypper lr -d` 显示的软件包储存库的 URL。
请对每个储存库使用一个 `--add-repo` 参数。

5. 在 DVD 或 USB 记忆棒上刻录 ISO。使用此类媒体引导要转换的计算机。
6. 系统启动后，会要求您提供转换服务器的连接细节。此服务器是装有 `virt-v2v` 软件包的计算机。
如果网络设置比 DHCP 客户端更复杂，请单击配置网络按钮打开 YaST 网络配置对话框。
单击测试连接按钮以转到向导的下一页。
7. 选择要转换的磁盘和网络接口，并定义 VM 数据，例如分配的 CPU 数量和内存量，以及虚拟机名称。



注意

如果不定义这些数据，创建的磁盘映像默认将采用 `raw` 格式。可以在输出格式字段中输入所需格式来更改此格式。

可通过两种方法生成虚拟机：使用 `local` 或 `libvirt` 输出。第一种方法会将虚拟机磁盘映像和配置放到输出储存字段中定义的路径。然后，可以通过 `virsh` 使用这些内容来定义由 `libvirt` 处理的新 Guest。第二种方法将使用放在输出储存字段所定义的池中的磁盘映像，来创建由 `libvirt` 处理的新 Guest。

单击开始转换开始该过程。

19.4 查错

19.4.1 Btrfs 相关的问题

对包含 Btrfs 根分区（在 SUSE Linux Enterprise Server 中为默认设置）的映像使用 guestfs 工具时，以下错误消息可能会显示：

```
tux > 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-file systems'. 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 file system(s) you want to see by manually adding '-m' option(s). Use 'virt-file systems' 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).

发生此问题通常是因为 Guest 中存在多个快照。在此情况下，guestfs 不知道要引导哪个快照。要强制使用某个快照，请按如下所示使用 `-m` 参数：

```
tux > virt-ls -m /dev/sda2::subvol=@/.snapshots/2/snapshot -a /path/to/
sles12sp2.qcow2 /
```

19.4.2 环境

在 libguestfs 设备中对问题进行查错时，可以使用环境变量 `LIBGUESTFS_DEBUG=1` 来启用调试消息。要以类似于 guestfish 命令的格式输出每个命令/API 调用，请使用环境变量 `LIBGUESTFS_TRACE=1`。

19.4.3 libguestfs-test-tool

libguestfs-test-tool 是一个测试程序，用于检查基本 libguestfs 功能是否正常工作。它会列显 guestfs 环境的大量诊断消息和细节，然后创建一个测试映像并尝试将其启动。如果它成功运行到完成时，则测试将近结束时应该会显示以下消息：

```
===== TEST FINISHED OK =====
```

19.5 外部参考信息

- libguestfs.org (<http://libguestfs.org>) ↗
- [libguestfs FAQ \(libguestfs 常见问题\)](http://libguestfs.org/guestfs-faq.1.html) (<http://libguestfs.org/guestfs-faq.1.html>) ↗

20 QEMU Guest 代理

QEMU Guest 代理 (GA) 在 VM Guest 中运行，使 VM 主机服务器能够通过 `libvirt` 在 Guest 操作系统中运行命令。它支持许多功能 — 例如，获取有关 Guest 文件系统的细节、冻结和解冻文件系统，或者挂起或重引导 Guest。

QEMU GA 包含在 `qemu-guest-agent` 软件包中，默认会在 KVM 或 Xen 虚拟机上安装、配置并激活。

20.1 运行 QEMU GA 命令

QEMU GA 包含的许多本机命令没有直接对应的 `libvirt` 命令。请参见第 20.4 节 “更多信息” 查看完整列表。您可以使用 `libvirt` 的通用命令 `qemu-agent-command` 来运行所有 QEMU GA 命令：

```
virsh qemu-agent-command DOMAIN_NAME '{"execute":"QEMU_GA_COMMAND"}'
```

例如：

```
tux > 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
      },
      [...]
    ]
  }
}
```

20.2 需要 QEMU GA 的 `virsh` 命令

有多个 `virsh` 命令需要 QEMU GA 才能实现其功能。其中包括：

virsh guestinfo

从 Guest 的角度列显有关该 Guest 的信息。

virsh guestvcpus

从 Guest 的角度查询或更改虚拟 CPU 的状态。

virsh set-user-password

为 Guest 中的用户帐户设置口令。

virsh domfsinfo

显示正在运行的域中装入的文件系统列表。

virsh dompmsuspend

挂起正在运行的 Guest。

20.3 增强 libvirt 命令

如果在 Guest 中启用了 QEMU GA，一些 **virsh** 子命令在以代理模式运行时，其功能会得到增强。下面的列表仅包含部分此类子命令示例。有关完整列表，请参见 **virsh** 手册页并搜索 **agent** 字符串。

virsh shutdown --mode agent 和 virsh reboot --mode agent

这种关机或重引导方法类似于 ACPI 方法，可让 Guest 为下次运行保持干净状态。

virsh domfsfreeze 和 virsh domfsthaw

指示 Guest 将其文件系统保持静止状态：刷新缓存中的所有 I/O 操作并将卷保持一致状态，以便在重新装入卷时无需进行任何检查。

virsh setvcpus --guest

更改指派给 Guest 的 CPU 数量。

virsh domifaddr --source agent

在 QEMU GA 中查询 Guest 的 IP 地址。

virsh vcpucount --guest

从 Guest 的角度列显有关虚拟 CPU 计数的信息。

20.4 更多信息

- <https://www.qemu.org/docs/master/interop/qemu-ga-ref.html>  上提供了 QEMU GA 支持的命令的完整列表。
- **virsh** 手册页 (**`man 1 virsh`**) 包含支持 QEMU GA 界面的命令的说明。

IV 使用 Xen 管理虚拟机

- 21 设置虚拟机主机 199
- 22 虚拟网络 211
- 23 管理虚拟化环境 219
- 24 Xen 中的块设备 225
- 25 虚拟化：配置选项和设置 229
- 26 管理任务 239
- 27 XenStore：在域之间共享的配置数据库 248
- 28 使用 Xen 作为高可用性虚拟化主机 254

21 设置虚拟机主机

本节介绍如何设置并使用 SUSE Linux Enterprise Server 15 SP2 作为虚拟机主机。

Dom0 的硬件要求通常与 SUSE Linux Enterprise Server 操作系统的硬件要求相同。应该添加额外的 CPU、磁盘、内存和网络资源才能满足规划的所有 VM Guest 系统的需求。



提示：资源

请记住，如果 VM Guest 系统在更快的处理器上运行并且可以访问更多系统内存，其性能就会更好，这与物理机一样。

虚拟机主机要求安装多个软件包及其依赖项。要安装全部所需的软件包，请运行 YaST 软件管理，选择视图 > 模式，然后选择 Xen 虚拟机主机服务器进行安装。也可以在 YaST 中使用模块虚拟化 > 安装超级管理程序和工具来执行安装。

安装 Xen 软件后，重新启动计算机，并在引导屏幕上选择新添加的具有 Xen 内核的选项。

通过更新通道可使用更新。为了确保安装最新的更新，请在完成安装后运行 YaST 联机更新。

21.1 最佳实践和建议

在主机上安装并配置 SUSE Linux Enterprise Server 操作系统后，请实施以下最佳实践和建议：

- 如果该主机应始终作为 Xen 主机运行，请运行 YaST 系统 > 引导加载程序，并激活 Xen 引导项作为默认引导部分。
 - 在 YaST 中单击系统 > 引导加载程序。
 - 将默认引导更改为 Xen 标签，然后单击设置为默认值。
 - 单击完成。
- 为获得最佳性能，请仅在虚拟机主机上安装虚拟化所需的应用程序和进程。

- 同时使用 iSCSI 和 OCFS2 托管 Xen 映像时，可能无法满足 SUSE Linux Enterprise Server 中的 OCFS2 默认超时所要求的延迟时间。要重新配置此超时，请运行 `systemctl configure o2cb`，或编辑系统配置中的 `O2CB_HEARTBEAT_THRESHOLD`。
- 如果您打算使用挂接到 Xen 主机的看门狗设备，请每次仅使用一个设备。建议使用提供实际硬件集成的驱动程序，而不要使用通用软件驱动程序。



注意：硬件监视

Dom0 内核以虚拟化模式运行，因此 `irqbalance` 或 `lscpu` 等工具不会反映真实的硬件特征。

21.2 管理 Dom0 内存

在以前的 SUSE Linux Enterprise Server 版本中，Xen 主机的默认内存分配模式是将所有主机物理内存都分配给 Dom0，并启用自动气球式调节功能。当有其他域启动后，内存会自动从 Dom0 进行气球式调节。此行为总是容易出错，因此强烈建议将其禁用。从 SUSE Linux Enterprise Server 15 SP1 开始，默认已禁用自动气球式调节，而是为 Dom0 分配 10% 的主机物理内存外加 1GB。例如，在物理内存大小为 32 GB 的主机上，将为 Dom0 分配 4.2 GB 内存。我们仍支持并建议在 `/etc/default/grub` 中使用 `dom0_mem` Xen 命令行选项。您可以通过将 `dom0_mem` 设置为主机物理内存大小，并在 `/etc/xen/xl.conf` 中启用 `autoballoon` 选项，来恢复旧行为。



警告：Dom0 内存不足

为 Dom0 保留的内存量取决于主机上运行的 VM Guest 数量，因为 Dom0 会为每个 VM Guest 提供后端网络和磁盘 I/O 服务。计算 Dom0 内存分配时，还应考虑到 Dom0 中运行的其他工作负载。一般而言，应该像确定任何其他虚拟机的内存大小一样来确定 Dom0 的内存大小。

21.2.1 设置 Dom0 内存分配

1. 确定需要为 Dom0 分配的内存。

2. 在 Dom0 中，键入 `xl info` 以查看计算机上可用的内存量。可以使用 `xl list` 命令确定当前为 Dom0 分配的内存。
3. 运行 YaST › 引导加载程序。
4. 选择 Xen 部分。
5. 在其他 Xen 超级管理程序参数中添加 `dom0_mem=MEM_AMOUNT`，其中 `MEM_AMOUNT` 是分配给 Dom0 的最大内存量。添加 `K`、`M` 或 `G` 来指定大小，例如 `dom0_mem=2G`。
6. 重新启动计算机以应用更改。



警告：Xen Dom0 内存

对 GRUB 2 中的 Xen 超级管理程序使用 XL 工具堆栈和 `dom0_mem=` 选项时，需在 `etc/xen/xl.conf` 中禁用 `xl autoballoon`。否则，启动 VM 时将会失败并出现有关无法压缩 Dom0 内存气球的错误。因此，如果您为 Xen 指定了 `dom0_mem=` 选项，请在 `xl.conf` 中添加 `autoballoon=0`。另请

参见 [Xen dom0 memory \(Xen dom0 内存\)](http://wiki.xen.org/wiki/Xen_dom0_memory) ([http://wiki.xen.org/wiki/](http://wiki.xen.org/wiki/Xen_dom0_memory)

[Xen_Best_Practices#Xen_dom0_dedicated_memory_and_preventing_dom0_memory_balloonin](http://wiki.xen.org/wiki/Xen_Best_Practices#Xen_dom0_dedicated_memory_and_preventing_dom0_memory_balloonin)

21.3 全虚拟化 Guest 中的网卡

在全虚拟化 Guest 中，默认网卡是一个模拟的 Realtek 网卡。不过，您也可以使用分离式网络驱动程序来管理 Dom0 与 VM Guest 之间的通讯。默认情况下，这两个接口都会呈现给 VM Guest，因为有些操作系统的驱动程序要求这两个接口都存在。

使用 SUSE Linux Enterprise Server 时，默认只有半虚拟化网卡可供 VM Guest 使用。可用的网络选项如下：

模拟

要使用模拟 Realtek 网卡之类的模拟网络接口，请在域 xl 配置的 `vif` 设备部分指定 `type=ioemu`。示例配置如下所示：

```
vif = [ 'type=ioemu,mac=00:16:3e:5f:48:e4,bridge=br0' ]
```

`xl.conf` 手册页 `man 5 xl.conf` 中提供了有关 xl 配置的更多细节。

半虚拟化

如果指定 `type=vif` 但不指定型号或类型，将使用半虚拟化网络接口：

```
vif = [ 'type=vif,mac=00:16:3e:5f:48:e4,bridge=br0,backen=0' ]
```

模拟和半虚拟化

如果要为管理员提供上述两个选项，只需同时指定类型和型号即可。xl 配置如下所示：

```
vif = [ 'type=ioemu,mac=00:16:3e:5f:48:e4,model=rtl8139,bridge=br0' ]
```

在这种情况下，应在 VM Guest 上禁用其中一个网络接口。

21.4 启动虚拟机主机

如果正确安装了虚拟化软件，计算机引导时会显示 GRUB 2 引导加载程序，且其菜单中会包含 Xen 选项。选择此选项即可启动虚拟机主机。



注意：Xen 和 Kdump

在 Xen 中，超级管理程序会管理内存资源。如果您需要为 Dom0 中的恢复内核保留系统内存，需由超级管理程序保留此内存。因此，需要将参数 `crashkernel=size` 添加到 `kernel` 行，而不能在该行中使用其他引导参数。

有关 `crashkernel` 参数的详细信息，请参见《系统分析和微调指南》，第 17 章 “Kexec 和 Kdump”，第 17.4 节 “计算 `crashkernel` 分配大小”。

如果 GRUB 2 菜单中不包含 Xen 选项，请检查安装步骤，并校验是否已更新 GRUB 2 引导加载程序。如果安装是在未选择 Xen 模式的情况下完成的，请运行 YaST 软件管理，选择模式过滤器，然后选择 Xen 虚拟机主机服务器进行安装。

引导超级管理程序后，Dom0 虚拟机将会启动并显示其图形桌面环境。如果您未安装图形桌面，则会显示命令行环境。



提示：图形问题

有时可能会发生图形系统无法正常工作的问题。在这种情况下，请将 `vga=ask` 添加到引导参数。要激活永久设置，请使用 `vga=mode-0x???`，其中 `???` 的计算方式为 `0x100 + http://en.wikipedia.org/wiki/VESA_BIOS_Extensions` 中所述的 VESA 模式，例如 `vga=mode-0x361`。

在开始安装虚拟 Guest 之前，请确保系统时间正确。为此，请在控制域上配置 NTP（网络时间协议）：

1. 在 YaST 中选择网络服务 > NTP 配置。
2. 选择在引导期间自动启动 NTP 守护程序的选项。提供现有 NTP 时间服务器的 IP 地址，然后单击完成。



注意：虚拟 Guest 上的时间服务

硬件时钟通常都不是非常精确。所有新式操作系统都会尝试通过一个额外的时间源来更正系统时间（对比硬件时间）。要使所有 VM Guest 系统上的时间正确，请也在每个相应 Guest 上激活网络时间服务，或确保 Guest 使用主机的系统时间。有关 SUSE Linux Enterprise Server 中的 独立时钟 的详细信息，请参见第 18.2 节 “Xen 虚拟机时钟设置”。

有关管理虚拟机的详细信息，请参见第 23 章 “管理虚拟化环境”。

21.5 PCI 直通

为了充分利用 VM Guest 系统，有时需要将特定的 PCI 设备指派给专用的域。如果使用的是全虚拟化 Guest，仅当系统的芯片组支持并且已在 BIOS 中激活此功能时，此功能才可用。

AMD* 和 Intel* 都提供此功能。对于 AMD 计算机，此功能称为 **IOMMU**；对于 Intel 而言，此功能称为 **VT-d**。请注意，具备 Intel-VT 技术还不足以对全虚拟化 Guest 使用此功能。为确保您的计算机支持此功能，需专门要求您的供应商提供一个支持 PCI 直通的系统。

限制

- 有些图形驱动程序使用高度优化的方法来访问 DMA。这种方法不受支持，因此使用显卡可能会存在问题。
- 访问 **PCIe** 网桥后面的 PCI 设备时，必须将所有 PCI 设备都指派到单个 Guest。这项限制不适用于 **PCIe** 设备。
- 具有专用 PCI 设备的 Guest 无法实时迁移到其他主机。

PCI 直通的配置要兼顾两个方面。首先，在引导时必须告知超级管理程序应使某个 PCI 设备可供重新指派。其次，必须将该 PCI 设备指派给 VM Guest。

21.5.1 配置超级管理程序以使用 PCI 直通

1. 选择要重指派给 VM Guest 的设备。为此，请运行 **lspci -k**，并读取设备编号以及指派给该设备的原始模块的名称：

```
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
```

在本例中，PCI 编号为 (06:01.0)，相关的内核模块为 e1000e。

2. 指定模块依赖项以确保 xen_pciback 是用于控制设备的第一个模块。添加包含以下内容的 /etc/modprobe.d/50-e1000e.conf 文件：

```
install e1000e /sbin/modprobe xen_pciback ; /sbin/modprobe \
--first-time --ignore-install e1000e
```

3. 指示 xen_pciback 模块使用“hide”选项来控制设备。编辑或创建包含以下内容的 /etc/modprobe.d/50-xen-pciback.conf 文件：

```
options xen_pciback hide=(06:01.0)
```

4. 重新启动系统。

5. 使用以下命令检查该设备是否在可指派设备列表中

```
xl pci-assignable-list
```

21.5.1.1 通过 xl 进行动态指派

为了避免重新启动主机系统，您可以利用通过 xl 进行的动态指派来使用 PCI 直通。

首先确保 Dom0 中已装载 pciback 模块：

```
tux > sudo modprobe pciback
```

然后使用 **xl pci-assignable-add** 使设备可供指派。例如，要使设备 06:01.0 可供 Guest 使用，请运行以下命令：

```
tux > sudo xl pci-assignable-add 06:01.0
```

21.5.2 将 PCI 设备指派给 VM Guest 系统

可通过多种方法来使 PCI 设备专用于某个 VM Guest：

安装时添加该设备：

在安装期间，在配置文件中添加 pci 行：

```
pci=['06:01.0']
```

将 PCI 设备热插入到 VM Guest 系统

可以使用 xl 命令即时添加或去除 PCI 设备。要将编号为 06:01.0 的设备添加到名为 sles12 的 Guest，请使用：

```
xl pci-attach sles12 06:01.0
```

将 PCI 设备添加到 Xend

要将设备永久添加到 Guest，请在 Guest 配置文件中添加以下代码段：

```
pci = [ '06:01.0,power_mgmt=1,permissive=1' ]
```

将 PCI 设备指派给 VM Guest 后，Guest 系统必须负责处理此设备的配置和设备驱动程序。

21.5.3 VGA 直通

Xen 4.0 和更高版本支持在全虚拟化 VM Guest 上实现 VGA 图形适配器直通。Guest 可以全面控制提供高性能全 3D 和视频加速的图形适配器。

限制

- VGA 直通功能与 PCI 直通类似，因此也需要主板芯片组和 BIOS 提供 IOMMU（或 Intel VT-d）支持。
- 只有主图形适配器（打开计算机电源时使用的图形适配器）能够与 VGA 直通搭配使用。
- 仅支持对全虚拟化 Guest 使用 VGA 直通。不支持半虚拟 (PV) Guest。
- 不能在多个使用 VGA 直通的 VM Guest 之间共享显卡 — 显卡只能供一个 Guest 专用。

要启用 VGA 直通，请在全虚拟化 Guest 配置文件中添加以下设置：

```
gfx_passthru=1
pci=['yy:zz.n']
```

其中，yy:zz.n 是 Dom0 上使用 `lspci -v` 找到的 VGA 图形适配器的 PCI 控制器 ID。

21.5.4 查错

在某些情况下，安装 VM Guest 期间可能会出现問題。本节将介绍一些已知问题及其解决方法。

系统在引导期间挂起

软件 I/O 转换缓冲区会提前在引导进程中分配一大块低速内存。如果内存请求超出了缓冲区大小，通常就会导致引导进程挂起。要检查是否存在这种情况，请切换到控制台 10，并检查其输出是否包含如下所示的消息

```
kernel: PCI-DMA: Out of SW-IOMMU space for 32768 bytes at device
000:01:02.0
```

在这种情况下，您需要增加 `swiotlb` 的大小。在 Dom0 的命令行上添加 `swiotlb=VALUE`（其中 `VALUE` 指定为 slab 项数）。请注意，可以通过增大或减小该数字来找到适合计算机的最佳大小。



注意：为 PV Guest 启用 swiotlb

要在 PV Guest 上正常进行 PCI 设备的 DMA 访问，必须指定 `swiotlb=force` 内核参数。有关 IOMMU 和 `swiotlb` 选项的详细信息，请参见软件包 `kernel-source` 中的 `boot-options.txt` 文件。

21.5.5 更多信息

因特网上的一些资源提供了有关 PCI 直通的有趣信息：

- https://wiki.xenproject.org/wiki/VTd_HowTo ↗
- <http://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices/> ↗
- http://support.amd.com/TechDocs/48882_IOMMU.pdf ↗

21.6 USB 直通

可通过两种方法将单个主机 USB 设备直通到 Guest。第一种方法是使用模拟的 USB 设备控制器，第二种方法是使用 PVUSB。

21.6.1 标识 USB 设备

在将 USB 设备直通到 VM Guest 之前，需要先在 VM 主机服务器上标识该设备。使用 `lsusb` 命令列出主机系统上的 USB 设备：

```
root # 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
```

例如，要直通 Dell 鼠标，请以 `vendor_id:device_id` (0461:4d15) 格式指定设备标记，或以 `bus.device` (2.3) 格式指定总线地址。请记得去除前导零，否则 `xl` 会将数字解释为八进制值。

21.6.2 模拟的 USB 设备

在模拟的 USB 中，设备模型 (QEMU) 会向 Guest 呈现模拟的 USB 控制器。然后，将通过 Dom0 控制 USB 设备，而 USB 命令将在 VM Guest 与主机 USB 设备之间转换。此方法仅适用于全虚拟化域 (HVM)。

使用 `usb=1` 选项启用模拟的 USB 集线器。然后使用 `host:USBID` 在配置文件中指定设备列表中的设备以及其他模拟的设备。例如：

```
usb=1
usbdevice=['tablet','host:2.3','host:0424:460']
```

21.6.3 半虚拟化 PVUSB

PVUSB 是以高性能方式实现从 Dom0 到虚拟化 Guest 的 USB 直通的新方法。借助 PVUSB，可以通过两种方式将 USB 设备添加到 Guest：

- 在创建域时通过配置文件添加
- 当 VM 正在运行时通过热插入方式添加

PVUSB 使用半虚拟化前端和后端接口。PVUSB 支持 USB 1.1 和 USB 2.0，适用于 PV 和 HVM Guest。要使用 PVUSB，Guest 操作系统中需要有 USB 前端，并且 Dom0 中需要有 USB 后端或者 QEMU 中需要有 USB 后端。在 SUSE Linux Enterprise Server 上，QEMU 已随附了 USB 后端。

从 Xen 4.7 开始引入了 `xl` PVUSB 支持和热插入支持。

在配置文件中，使用 `usbctrl` 和 `usbdev` 指定 USB 控制器和 USB 主机设备。例如，对于 HVM Guest：


```
usbctrl=['type=qusb,version=2,ports=4', 'type=qusb,version=1,ports=4', ]
usbdev=['hostbus=2, hostaddr=1, controller=0,port=1', ]
```



注意

必须为 HVM Guest 的控制器指定 `type=qusb`。

要管理 PVUSB 设备的热插入，请使用 `usbctrl-attach`、`usbctrl-detach`、`usb-list`、`usbdev-attach` 和 `usb-detach` 子命令。例如：

创建版本为 USB 1.1 并包含 8 个端口的 USB 控制器：

```
root # xl usbctrl-attach test_vm version=1 ports=8 type=qusb
```

在域中找到第一个可用的 controller:port，并将 busnum:devnum 为 2:3 的 USB 设备挂接到该端口；您也可以指定 控制器 和 端口：

```
root # xl usbdev-attach test_vm hostbus=2 hostaddr=3
```

显示域中的所有 USB 控制器和 USB 设备：

```
root # 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:
```

分离控制器 0 端口 1 下的 USB 设备：

```
root # xl usbdev-detach test_vm 0 1
```

去除具有所示 dev_id 的 USB 控制器，以及其下的所有 USB 设备：

```
root # xl usbctrl-detach test_vm dev_id
```

有关更多信息，请参见https://wiki.xenproject.org/wiki/Xen_USB_Passthrough 。

22 虚拟网络

VM Guest 系统需要通过某种方式才能与其他 VM Guest 系统或本地网络通讯。用于连接 VM Guest 系统的网络接口由一个分离式设备驱动程序构成，也就是说，任何虚拟以太网设备在 Dom0 中都有一个对应的网络接口。此接口设置为访问 Dom0 中运行的虚拟网络。SUSE Linux Enterprise Server 的系统配置中全面集成了桥接式虚拟网络，您可以通过 YaST 配置该网络。

安装 Xen VM 主机服务器时，系统会在常规网络配置期间建议一种桥接式网络配置。用户可以选择在安装期间更改配置，并可根据本地需求对其进行自定义。

如果需要，可以在进行默认的物理服务器安装后，使用 YaST 中的 [安装超级管理程序和工具](#) 模块来安装 Xen VM 主机服务器。此模块会使系统做好托管虚拟机的准备，包括调用默认网桥功能的建议。

如果使用 [rpm](#) 或 [zypper](#) 手动安装了 Xen VM 主机服务器所需的软件包，则其余的系统配置需要由管理员手动完成，或者通过 YaST 完成。

SUSE Linux Enterprise Server 中默认不使用 Xen 提供的网络脚本。这些脚本仅供参考，并且已禁用。SUSE Linux Enterprise Server 中使用的网络配置通过 YaST 系统配置完成，该系统配置类似于 SUSE Linux Enterprise Server 中的网络接口配置。

有关管理网桥的更多一般信息，请参见 [第 13.1 节 “网桥”](#)。

22.1 Guest 系统的网络设备

Xen 超级管理程序可以提供不同类型的网络接口来连接 VM Guest 系统。首选网络设备应该是半虚拟化网络接口。此类网络接口的系统要求最低，却能实现最高的传输速率。最多可为每个 VM Guest 提供八个网络接口。

无法感知半虚拟化硬件的系统可能无法使用此方案。有多个模拟网络接口可用于将系统连接到只能以全虚拟化模式运行的网络。您可以自行选用以下模拟产品：

- Realtek 8139 (PCI)。这是默认的模拟网卡。
- AMD PCnet32 (PCI)
- NE2000 (PCI)
- NE2000 (ISA)

- Intel e100 (PCI)
- Intel e1000 及其衍生产品 e1000-82540em、e1000-82544gc 和 e1000-82545em (PCI)

以上网络接口全部都是软件接口。由于每个网络接口都必须具有唯一的 MAC 地址，因此已向这些接口可以使用的 Xensource 指派了一个地址范围。



提示：虚拟网络接口和 MAC 地址

虚拟化环境中的默认 MAC 地址配置会创建类似于 00:16:3E:xx:xx:xx 的随机 MAC 地址。一般情况下，可用 MAC 地址的数量应该足够大才能获得唯一地址。但是，如果您的安装规模非常大，或者要确保随机 MAC 地址指派不会造成问题，您也可以手动指派这些地址。

进行调试或系统管理时，知道 Dom0 中的哪个虚拟接口连接到正在运行的 Guest 中的哪个以太网设备可能很有帮助。可以从 Dom0 中的设备名称获得此信息。所有虚拟设备都遵循 `vif<domain number>.<interface_number>` 这样的命名规则。

例如，如果您想知道 ID 为 5 的 VM Guest 的第三个接口 (eth2) 的设备名称，那么 Dom0 中的该设备应该是 `vif5.2`。要获取所有可用接口的列表，请运行 `ip a` 命令。

设备名称不包含有关此接口连接到哪个网桥的任何信息，但 Dom0 中提供了此信息。要大致了解有关哪个接口连接到哪个网桥的信息，请运行 `bridge link` 命令。输出可能如下所示：

```
tux > 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
```

此示例中配置了三个网桥：`br0`、`br1` 和 `br2`。当前为 `br0` 和 `br1` 各添加了一个真实的以太网设备，分别是 `eth0` 和 `eth1`。

22.2 Xen 中基于主机的路由

您可将 Xen 设置为在控制 Dom0 中使用基于主机的路由。遗憾的是，YaST 目前还不能很好地支持此设置，需要手动对配置文件进行相当多的编辑才能使用此设置。因此，此任务需要由高级管理员来完成。

仅当使用固定 IP 地址时，以下配置才起作用。在此过程中不能使用 DHCP，因为 VM Guest 和 VM 主机服务器系统都必须知道 IP 地址。

要创建路由式 Guest，最简单的方法就是将网络从桥接网络更改为路由网络。要执行以下过程，必须先安装一个使用桥接网络的 VM Guest。例如，VM 主机服务器名为 earth，IP 为 192.168.1.20；VM Guest 名为 alice，IP 为 192.168.1.21。

过程 22.1：配置路由式 IPV4 VM GUEST

1. 确保 alice 已关机。使用 `xl` 命令关机并检查。
2. 在 VM 主机服务器 earth 上准备网络配置：
 - a. 创建一个用于路由流量的热插拔接口。为此，请创建包含以下内容的 `/etc/sysconfig/network/ifcfg-alice.0` 文件：

```
NAME="Xen guest alice"
BOOTPROTO="static"
STARTMODE="hotplug"
```

- b. 确保已启用 IP 转发：
 - i. 在 YaST 中，转到网络设置 > 路由。
 - ii. 进入路由选项卡，激活启用 IPv4 转发和启用 IPv6 转发选项。
 - iii. 确认设置并退出 YaST。
 - c. 将以下配置应用于 `firewalld`：

- 将 alice.0 添加到公共区域中的设备：

```
tux > sudo firewall-cmd --zone=public --add-interface=alice.0
```

- 告知防火墙要转发哪个地址：

```
tux > sudo firewall-cmd --zone=public \
--add-forward-
port=port=80:proto=tcp:toport=80:toaddr="192.168.1.21/32,0/0"
```

- 使运行时配置更改永久生效：

```
tux > sudo firewall-cmd --runtime-to-permanent
```

- d. 将一个静态路由添加到 alice 的接口。为此，请将下面一行添加到 /etc/sysconfig/network/routes 的末尾：

```
192.168.1.21 - - alice.0
```

- e. 为确保 VM 主机服务器连接的交换机和路由器知道路由接口，请在 earth 上激活 proxy_arp。将下面几行添加到 /etc/sysctl.conf 中：

```
net.ipv4.conf.default.proxy_arp = 1
net.ipv4.conf.all.proxy_arp = 1
```

- f. 使用以下命令激活所有更改：

```
tux > sudo systemctl restart systemd-sysctl wicked
```

3. 按照第 23.1 节 “XL — Xen 管理工具” 中所述，通过更改 alice 的 vif 接口配置继续完成 VM Guest 的 Xen 配置。对配置过程中生成的文本文件进行以下更改：

- a. 去除以下代码段

```
bridge=br0
```

- b. 添加以下代码段：

```
vifname=vifalice.0
```

或

```
vifname=vifalice.0=emu
```

（针对全虚拟化域）。

- c. 按如下所示更改用于设置接口的脚本：

```
script=/etc/xen/scripts/vif-route-ifup
```

- d. 激活新配置并启动 VM Guest。
4. 其余配置任务必须在 VM Guest 内部完成。
 - a. 使用 `xl console DOMAIN` 打开与 VM Guest 的控制台，然后登录。
 - b. 检查 Guest IP 是否设置为 192.168.1.21。
 - c. 为 VM Guest 提供用于连接 VM 主机服务器的主机路由和默认网关。为此，请将下面几行添加到 `/etc/sysconfig/network/routes` 中：

```
192.168.1.20 - - eth0
default 192.168.1.20 - -
```
5. 最后，测试从 VM Guest 到外部世界的网络连接，以及从网络到 VM Guest 的连接。

22.3 创建掩蔽网络设置

创建掩蔽网络设置与创建路由设置的过程非常相似。不过，创建掩蔽网络设置时不需要 `proxy_arp`，并且某些防火墙规则不同。要为 IP 地址为 192.168.100.1 并且其主机在 `br0` 上有自己的外部接口的 Guest（名为 dolly）创建掩蔽网络，请执行以下操作。为了简化配置，此处仅将已安装的 Guest 修改为使用掩蔽网络：

过程 22.2：配置掩蔽式 IPV4 VM GUEST

1. 使用 `xl shutdown DOMAIN` 关闭 VM Guest 系统。
2. 在 VM 主机服务器上准备网络配置：
 - a. 创建一个用于路由流量的热插拔接口。为此，请创建包含以下内容的 `/etc/sysconfig/network/ifcfg-dolly.0` 文件：

```
NAME="Xen guest dolly"
BOOTPROTO="static"
STARTMODE="hotplug"
```
 - b. 编辑文件 `/etc/sysconfig/SuSEfirewall2` 并添加以下配置：

- 将 dolly.0 添加到 FW_DEV_DMZ 中的设备：

```
FW_DEV_DMZ="dolly.0"
```

- 在防火墙中打开路由：

```
FW_ROUTE="yes"
```

- 在防火墙中打开掩蔽：

```
FW_MASQUERADE="yes"
```

- 告知防火墙要掩蔽哪个网络：

```
FW_MASQ_NETS="192.168.100.1/32"
```

- 从掩蔽例外中去除网络：

```
FW_NOMASQ_NETS=""
```

- 最后，使用以下命令重新启动防火墙：

```
tux > sudo systemctl restart SuSEfirewall2
```

- c. 向 dolly 的接口添加一个静态路由。为此，请将下面一行添加到 /etc/sysconfig/network/routes 的末尾：

```
192.168.100.1 - - dolly.0
```

- d. 使用以下命令激活所有更改：

```
tux > sudo systemctl restart wicked
```

3. 继续完成 VM Guest 的 Xen 配置。

- 按照第 23.1 节 “XL — Xen 管理工具” 中所述更改 dolly 的 vif 接口配置。
- 去除下面一项：


```
bridge=br0
```

- c. 添加以下代码段：

```
vifname=vifdolly.0
```

- d. 按如下所示更改用于设置接口的脚本：

```
script=/etc/xen/scripts/vif-route-ifup
```

- e. 激活新配置并启动 VM Guest。

4. 其余配置任务需在 VM Guest 内部完成。

- a. 使用 **xl console DOMAIN** 打开与 VM Guest 的控制台，然后登录。

- b. 检查 Guest IP 是否设置为 192.168.100.1。

- c. 为 VM Guest 提供用于连接 VM 主机服务器的主机路由和默认网关。为此，请将下面几行添加到 `/etc/sysconfig/network/routes` 中：

```
192.168.1.20 - - eth0
default 192.168.1.20 - -
```

5. 最后，测试从 VM Guest 到外部世界的网络连接。

22.4 特殊注意事项

在 Xen 中可以使用许多可行的网络配置。以下配置默认未激活：

22.4.1 虚拟网络中的带宽限制

在 Xen 中，您可以限制虚拟 Guest 在访问网桥时可使用的网络传输速率。要配置该限制，需要按照第 23.1 节“[XL — Xen 管理工具](#)”中所述修改 VM Guest 配置。

在配置文件中，先搜索连接到虚拟网桥的设备。配置如下所示：

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0' ]
```

要添加最大传输速率，请按如下所示在配置中添加参数 `rate`：

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0,rate=100Mb/s' ]
```

请注意，速率单位为 `Mb/s`（每秒兆位数）或 `MB/s`（每秒兆字节数）。在上面的示例中，虚拟接口的最大传输速率为 100 兆位。默认情况下，Guest 与虚拟网桥之间的带宽没有限制。

您甚至可以通过指定用于定义信用补充粒度的时段来微调该行为：

```
vif = [ 'mac=00:16:3e:4f:94:a9,bridge=br0,rate=100Mb/s@20ms' ]
```

22.4.2 监视网络流量

要监视特定接口上的流量，不妨使用一个实用的小应用程序 `iftop`，它可在终端中显示当前网络流量。

运行 Xen VM 主机服务器时，需要定义受监视接口。Dom0 用来访问物理网络的接口是网桥设备，例如 `br0`。但在您的系统上可能不是这样。要监视发往物理接口的所有流量，请以 `root` 身份运行终端并使用以下命令：

```
iftop -i br0
```

要监视特定 VM Guest 的特殊网络接口的网络流量，请提供正确的虚拟接口。例如，要监视 ID 为 5 的域的第一个以太网设备，请使用以下命令：

```
ftop -i vif5.0
```

要退出 `iftop`，请按 `q` 键。手册页 `man 8 iftop` 中介绍了更多选项和可行做法。

23 管理虚拟化环境

除了使用建议的 `libvirt` 库（第 II 部分 “使用 `libvirt` 管理虚拟机”）以外，您还可以在命令行中使用 `xl` 工具来管理 Xen Guest 域。

23.1 XL — Xen 管理工具

`xl` 程序是用于管理 Xen Guest 域的工具。它包含在 `xen-tools` 软件包中。`xl` 基于 `LibXenlight` 库，可用于执行一般的域管理工作，例如创建、监听、暂停或关闭域。通常只有 `root` 用户才能执行 `xl` 命令。



注意

`xl` 只能管理域配置文件指定的运行中 Guest 域。如果某个 Guest 域未运行，则您无法使用 `xl` 来管理它。



提示

为了允许用户继续像使用已过时的 `xm` 命令那样来使用受管 Guest 域，目前我们建议使用 `libvirt` 的 `virsh` 和 `virt-manager` 工具。有关更多信息，请参见第 II 部分 “使用 `libvirt` 管理虚拟机”。

`xl` 操作依赖于 `xenstored` 和 `xenconsole` 服务。请确保在引导时启动

```
tux > systemctl start xencommons
```

以初始化 `xl` 所需的所有守护程序。



提示：在主机域中设置 `xenbr0` 网桥

在最常用的网络配置中，需在主机域中设置一个名为 `xenbr0` 的网桥，以便为 Guest 域提供正常工作的网络。

每个 `xl` 命令的基本结构如下：

```
xl <subcommand> [options] domain_id
```

其中，<subcommand> 是要运行的 xl 命令，domain_id 是指派给域的 ID 编号或虚拟机的名称，**OPTIONS** 表示特定于子命令的选项。

如需可用 **xl** 子命令的完整列表，请运行 **xl help**。对于每个命令，您都可以使用附加参数 **--help** 获取更详细的帮助。**xl** 的手册页中提供了有关相应子命令的详细信息。

例如，**xl list --help** 会显示 list 命令可用的所有选项。举例来说，**xl list** 命令可显示所有虚拟机的状态。

```
tux > 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 信息指示某个计算机是否正在运行，以及处于哪种状态。最常用的标志为 **r**（正在运行）和 **b**（受阻），其中“受阻”的意思是该计算机正在等待 IO，或者由于无需执行任何操作而处于休眠状态。有关状态标志的更多细节，请参见 **man 1 xl**。

其他有用的 **xl** 命令包括：

- **xl create**，可基于给定的配置文件创建虚拟机。
- **xl reboot**，可重引导虚拟机。
- **xl destroy**，可立即终止虚拟机。
- **xl block-list**，可显示挂接到虚拟机的所有虚拟块设备。

23.1.1 Guest 域配置文件

使用 **xl** 操作域时，每个域都需有相应的域配置文件。用于储存此类配置文件的默认目录为 **/etc/xen/**。

域配置文件是一个纯文本文件。它包含多个“**键 = 值**”对。有些键是必需的，有些键是通用的且适用于任何 Guest，还有些键只适用于特定的 Guest 类型（半虚拟化或全虚拟化）。值可以是括在单引号或双引号中的字符串（**"string"** 形式）、数字、布尔值，或者括在方括号中的多个值的列表（**[value1, value2, ...]** 形式）。

例 23.1：SLED 12 的 GUEST 域配置文件：/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"
```

要启动此类域，请运行 **xl create /etc/xen/sled12.cfg**。

23.2 自动启动 Guest 域

要使 Guest 域在主机系统引导后自动启动，请执行以下步骤：

1. 创建域配置文件（如果不存在）并将其保存到 /etc/xen/ 目录，例如 /etc/xen/domain_name.cfg。
2. 在 auto/ 子目录中创建 Guest 域配置文件的符号链接。

```
tux > sudo ln -s /etc/xen/domain_name.cfg /etc/xen/auto/domain_name.cfg
```

3. 系统下次引导时，domain_name.cfg 中定义的 Guest 域将会启动。

23.3 事件操作

在 Guest 域配置文件中，您可以定义在发生一组预定义的事件时要执行的操作。例如，要告知域在其关机后自行重启动，请在其配置文件中包含下面一行：

```
on_poweroff="restart"
```

下面是 Guest 域的预定义事件列表：

事件列表

on_poweroff

指定在域自行关机后应执行什么操作。

on_reboot

当域关机并提供了请求重引导的原因代码时要执行的操作。

on_watchdog

当域由于 Xen 看门狗超时而关机时要执行的操作。

on_crash

当域崩溃时要执行的操作。

对于这些事件，可以定义以下操作之一：

相关操作列表

destroy

销毁域。

restart

销毁域，并立即采用相同的配置创建新域。

rename-restart

重命名已终止的域，然后立即采用与原始域相同的配置创建新域。

preserve

保留域。可以检查该域，以后再使用 **`xl destroy`** 将它销毁。

coredump-destroy

将域的核心转储写入 `/var/xen/dump/NAME`，然后销毁该域。

coredump-restart

将域的核心转储写入 `/var/xen/dump/NAME`，然后重新启动该域。

23.4 时戳计数器

您可以为 Guest 域配置文件中的每个域指定时戳计数器 (TSC)（有关详细信息，请参见第 23.1.1 节“Guest 域配置文件”）。

使用 **`tsc_mode`** 设置可以指定是要“本机”执行 `rdtsc` 指令（速度较快，但 TSC 敏感型应用程序有时无法正常运行），还是模拟这些指令（始终可正常运行，但性能可能受到影响）。

tsc_mode=0（默认设置）

使用此设置可确保正常运行，同时提供可以实现的最佳性能 — 有关详细信息，请参见 <https://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt> 。

tsc_mode=1（始终模拟）

如果 TSC 敏感型应用程序正在运行，并且已知且可接受最坏情况下的性能下降，请使用此设置。

tsc_mode=2（永不模拟）

如果此 VM 中运行的所有应用程序都能够灵活适应 TSC 并且需要最高性能，请使用此设置。

tsc_mode=3（PVRDTSCP）

可以半虚拟化（修改）高 TSC 频率应用程序，以同时兼顾正常运行和最高性能 — 任何未经修改的应用程序都必须能够灵活适应 TSC。

有关背景信息，请参见 <https://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt> 。

23.5 保存虚拟机

过程 23.1：保存虚拟机的当前状态

1. 确保要保存的虚拟机正在运行。
2. 在主机环境中输入

```
tux > sudo xl save ID STATE-FILE
```

其中，ID 是要保存的虚拟机 ID，STATE-FILE 是您为内存状态文件指定的名称。默认情况下，当您创建域的快照后，该域将不再运行。使用 -c 可使域保持运行状态，即使在您创建快照后也是如此。

23.6 恢复虚拟机

过程 23.2：恢复虚拟机的当前状态

1. 确保要恢复的虚拟机自您运行保存操作后始终未启动。
2. 在主机环境中输入

```
tux > sudo xl restore STATE-FILE
```

其中，STATE-FILE 是先前保存的内存状态文件。域在恢复后默认将处于运行状态。要在恢复后暂停域，请使用 -p。

23.7 虚拟机状态

可以通过查看 xl list 命令的结果来显示虚拟机的状态，结果中以单字符缩写形式显示状态。

- r - 正在运行 - 虚拟机当前正在运行，并在消耗分配的资源。
- b - 受阻 - 虚拟机的处理器未运行，且无法运行。虚拟机正在等待 I/O 或已停止工作。
- p - 已暂停 - 虚拟机已暂停。虚拟机不会与超级管理程序交互，但仍保有其分配的资源，例如内存。
- s - 已关闭 - Guest 操作系统正在关闭、已重引导或已挂起，并且虚拟机正在停止。
- c - 已崩溃 - 虚拟机已崩溃，未在运行。
- d - 即将死机 - 虚拟机正在关机或崩溃。

24 Xen 中的块设备

24.1 将物理储存映射到虚拟磁盘

域配置文件中 Xen 域的磁盘指定方式非常直接，如下面的示例所示：

```
disk = [ 'format=raw,vdev=hdc,access=ro,devtype=cdrom,target=/root/image.iso' ]
```

此命令基于 `/root/image.iso` 磁盘映像文件定义了一个磁盘块设备。Guest 将该磁盘视为允许进行只读 (`ro`) 访问的 `hdc`。设备类型为 `cdrom`，格式为 `raw`。

下面的示例定义了相同的设备，但使用的是简化的位置语法：

```
disk = [ '/root/image.iso,raw,hdc,ro,cdrom' ]
```

您可在同一行中包含更多磁盘定义，每个定义需以逗号分隔。如果未指定某个参数，系统会采用其默认值：

```
disk = [ '/root/image.iso,raw,hdc,ro,cdrom', '/dev/vg/guest-volume,,hda','...' ]
```

参数列表

target

源块设备或磁盘映像文件路径。

format

映像文件的格式。默认值为 `raw`。

vdev

Guest 看到的虚拟设备。支持的值为 `hd[x]`、`xvd[x]`、`sd[x]` 等。有关更多细节，请参见 `/usr/share/doc/packages/xen/misc/vbd-interface.txt`。此参数是必需的。

access

提供给 Guest 的块设备处于只读模式还是读写模式。支持的值为 `ro` 或 `r`（只读访问）以及 `rw` 或 `w`（读写访问）。对于 `devtype=cdrom`，默认值为 `ro`；对于其他设备类型，默认值为 `rw`。

devtype

限定虚拟设备类型。支持的值为 `cdrom`。

backendtype

要使用的后端实现。支持的值为 `phy`、`tap` 和 `qdisk`。一般不应指定此选项，因为系统会自动确定后端类型。

script

指定 `target` 不是常规的主机路径，而是要由可执行程序解释的信息。如果指定的脚本文件不指向绝对路径，将在 `/etc/xen/scripts` 中查找该文件。这些脚本通常名为 `block-<script_name>`。

有关指定虚拟磁盘的详细信息，请参见 `/usr/share/doc/packages/xen/misc/xl-disk-configuration.txt`。

24.2 将网络储存映射到虚拟磁盘

与映射本地磁盘映像（请参见第 24.1 节“将物理储存映射到虚拟磁盘”）类似，您也可以将网络磁盘映射为虚拟磁盘。

下面的示例说明如何映射一个启用了多个 Ceph 监视器和 cephx 身份验证的 RBD（RADOS 块设备）：

```
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' ]
```

下面是 NBD（网络块设备）磁盘映射示例：

```
disk = [ 'vdev=hdc, backendtype=qdisk, target=nb:151.155.144.82:5555' ]
```

24.3 基于文件的虚拟磁盘和回写设备

当虚拟机正在运行时，其每个基于文件的虚拟磁盘都会占用主机上的一个回写设备。默认情况下，主机最多允许占用 64 个回写设备。

要在主机上同时运行更多基于文件的虚拟磁盘，可以通过将以下选项添加到主机的 `/etc/modprobe.conf.local` 文件，来增加可用回写设备的数量。

```
options loop max_loop=x
```

其中，`x` 是要创建的回写设备的最大数量。

重新装载模块后，更改即会生效。



提示

输入 `rmmod loop` 和 `modprobe loop` 可以卸载和重新装载模块。如果 `rmmod` 不起作用，请卸载所有现有循环设备，或重引导计算机。

24.4 调整块设备的大小

尽管您始终都可以向 VM Guest 系统添加新的块设备，但有时更合适的做法是增加现有块设备的大小。如果在部署 VM Guest 期间已经规划了此类系统修改，则应该注意一些基本事项：

- 使用可以增加大小的块设备。通常使用 LVM 设备和文件系统映像。
- 不要将 VM Guest 内部的设备分区，而是直接使用主设备来应用文件系统。例如，直接使用 `/dev/xvdb`，而不是向 `/dev/xvdb` 添加分区。
- 确保要使用的文件系统可调整大小。有时（例如，使用 Ext3 时），必须关闭某些功能才能调整文件系统的大小。其中一个可以联机调整大小并装入的文件系统是 `XFS`。增加底层块设备的大小后，使用 `xfs_growfs` 命令调整该文件系统的大小。有关 `XFS` 的详细信息，请参见 `man 8 xfs_growfs`。

调整指派给 VM Guest 的 LVM 设备的大小后，VM Guest 会自动获悉新的大小。您无需执行其他操作告知 VM Guest 块设备的新大小。

使用文件系统映像时，将使用一个循环设备向 Guest 挂接映像文件。有关调整该映像的大小以及为 VM Guest 刷新大小信息的详细信息，请参见第 26.2 节“稀疏映像文件和磁盘空间”。

24.5 用于管理高级储存方案的脚本

可以借助脚本来管理高级储存方案，例如 **dmmd**（“设备映射程序 — 多磁盘”）提供的磁盘环境，包括构建在软件 RAID 集基础之上的 LVM 环境，或者构建在 LVM 环境基础之上的软件 RAID 集。这些脚本包含在 **xen-tools** 软件包中。安装后，可以在 **/etc/xen/scripts** 中找到它们：

- **block-dmmd**
- **block-drbd-probe**
- **block-npiv**

借助这些脚本，您可以在将块设备提供给 Guest 之前使用外部命令执行这些设备的某项操作或一系列操作。

以前，只能使用磁盘配置语法 **script=** 将这些脚本与 **xl** 或 **libxl** 搭配使用。现在，可以通过在磁盘的 **<source>** 元素中指定块脚本的基本名称，将这些脚本与 libvirt 搭配使用。例如：

```
<source dev='dmmd:md;/dev/md0;lvm;/dev/vgxen/lv-vm01' />
```

25 虚拟化：配置选项和设置

本节中的内容介绍可帮助技术创新者实施前沿虚拟化解决方案的高级管理任务和配置选项。这些内容仅作为参考信息，并不意味着所述的所有选项和任务都受 Novell, Inc. 的支持。

25.1 虚拟 CD 读取器

可以在创建虚拟机时设置虚拟 CD 读取器，也可以将虚拟 CD 读取器添加到现有虚拟机。虚拟 CD 读取器可以基于物理 CD/DVD 或 ISO 映像。虚拟 CD 读取器的工作方式根据其是半虚拟还是全虚拟读取器而异。

25.1.1 半虚拟计算机上的虚拟 CD 读取器

一台半虚拟计算机最多可以包含 100 个由虚拟 CD 读取器和虚拟磁盘构成的块设备。在半虚拟计算机上，虚拟 CD 读取器会将 CD 显示为允许进行只读访问的虚拟磁盘。虚拟 CD 读取器不可用于向 CD 写入数据。

访问完半虚拟计算机上的 CD 后，建议从虚拟机中去除虚拟 CD 读取器。

半虚拟化 Guest 可以使用 `devtype=cdrom` 设备类型。这可以在一定程度上模拟真实 CD 读取器的行为，并允许更换 CD。您甚至可以使用 `eject` 命令打开 CD 读取器的托盘。

25.1.2 全虚拟计算机上的虚拟 CD 读取器

一台全虚拟计算机最多可以包含 4 个由虚拟 CD 读取器和虚拟磁盘构成的块设备。全虚拟计算机上的虚拟 CD 读取器会像物理 CD 读取器一样与插入的 CD 进行交互。

将 CD 插入主机计算机上的物理 CD 读取器后，包含基于物理 CD 读取器（例如 `/dev/cdrom/`）的虚拟 CD 读取器的所有虚拟机都可以读取插入的 CD。假设操作系统具有自动装入功能，CD 应该会自动显示在文件系统中。虚拟 CD 读取器不可用于向 CD 写入数据。它们配置为只读设备。

25.1.3 添加虚拟 CD 读取器

虚拟 CD 读取器可以基于插入 CD 读取器中的 CD，也可以基于 ISO 映像文件。

1. 请确保虚拟机正在运行，并且操作系统已完成引导。
2. 将所需 CD 插入物理 CD 读取器，或者将所需 ISO 映像复制到 Dom0 可访问的位置。
3. 在 VM Guest 中选择一个新的未使用的块设备，例如 `/dev/xvdb`。
4. 选择您要指派给 Guest 的 CD 读取器或 ISO 映像。
5. 使用真实 CD 读取器时，请使用以下命令将 CD 读取器指派给 VM Guest。在此示例中，Guest 名为 `alice`：

```
tux > sudo xl block-attach alice target=/dev/sr0,vdev=xvdb,access=ro
```

6. 指派映像文件时，请使用以下命令：

```
tux > sudo xl block-attach alice target=/path/to/  
file.iso,vdev=xvdb,access=ro
```

7. 新的块设备（例如 `/dev/xvdb`）即会添加到虚拟机中。
8. 如果虚拟机运行的是 Linux，请完成以下操作：
 - a. 在虚拟机中打开一个终端，然后输入 `fdisk -l` 校验是否正确添加了设备。您也可以输入 `ls /sys/block` 查看虚拟机可用的所有磁盘。
虚拟机将 CD 识别为带有驱动器号的虚拟磁盘，例如：

```
/dev/xvdb
```

- b. 输入使用相应驱动器号装入 CD 或 ISO 映像的命令。例如，

```
tux > sudo mount -o ro /dev/xvdb /mnt
```

会将 CD 装入到名为 `/mnt` 的安装点。

虚拟机应该可以通过指定的安装点使用该 CD 或 ISO 映像文件。

9. 如果虚拟机运行的是 Windows，请重引导虚拟机。

校验虚拟 CD 读取器是否显示在 [我的电脑](#) 部分。

25.1.4 去除虚拟 CD 读取器

1. 请确保虚拟机正在运行，并且操作系统已完成引导。
2. 如果已装入虚拟 CD 读取器，请从虚拟机内部将其卸载。
3. 在 Guest 块设备的主机视图输入 `xl block-list alice`。
4. 输入 `xl block-detach alice BLOCK_DEV_ID` 以从 Guest 中去除该虚拟设备。如果该操作失败，请尝试添加 `-f` 强制去除。
5. 按下硬件弹出按钮以弹出 CD。

25.2 远程访问方法

某些配置（例如包含机架式服务器的配置）要求运行的计算机不配备视频监视器、键盘或鼠标。此类配置通常称为 [无头](#) 配置，需要使用远程技术来管理。

典型的配置方案和技术包括：

包含 X Window Server 的图形桌面

如果在虚拟机主机上安装了图形桌面（例如 GNOME），您便可以使用 VNC 查看器这样的远程查看器。在远程计算机上，使用 `tigervnc` 或 `virt-viewer` 等图形工具登录并管理远程 Guest 环境。

仅限文本

您可以从远程计算机使用 `ssh` 命令登录到虚拟机主机并访问其基于文本的控制台。然后便可使用 `xl` 命令来管理虚拟机，以及使用 `virt-install` 命令来创建新虚拟机。

25.3 VNC 查看器

VNC 查看器用于以图形方式查看运行中 Guest 系统的环境。可以从 Dom0 使用该查看器（称为本地访问或机上访问），也可以从远程计算机使用。

可以使用 VM 主机服务器的 IP 地址和 VNC 查看器来查看此 VM Guest 的显示画面。当虚拟机运行时，主机上的 VNC 服务器将为该虚拟机指派一个端口号，用于建立 VNC 查看器连接。指派的端口号是虚拟机启动时可用的最小端口号。该端口号仅供虚拟机运行时使用。虚拟机关机后，该端口号可能会指派给其他虚拟机。

例如，如果端口 1、2、4、5 已指派给运行中的虚拟机，那么 VNC 查看器将指派最小可用端口号 3。如果虚拟机下次启动时端口号 3 仍在使用中，则 VNC 服务器将向虚拟机指派其他端口号。

要从远程计算机使用 VNC 查看器，防火墙必须允许访问 VM Guest 系统要通过其运行的端口数。即，要允许访问 5900 及更大编号的端口。例如，要运行 10 个 VM Guest 系统，则需要打开 TCP 端口 5900:5910。

要从运行 VNC 查看器客户端的本地控制台访问虚拟机，请输入以下命令之一：

- `vncviewer ::590#`
- `vncviewer :#`

`#` 是指派给虚拟机的 VNC 查看器端口号。

从 Dom0 之外的计算机访问 VM Guest 时，请使用以下语法：

```
tux > vncviewer 192.168.1.20::590#
```

在本例中，Dom0 的 IP 地址为 192.168.1.20。

25.3.1 向虚拟机指派 VNC 查看器端口号

尽管 VNC 查看器默认会指派第一个可用的端口号，但您应该向特定的虚拟机指派特定的 VNC 查看器端口号。

要在 VM Guest 上指派特定的端口号，请编辑虚拟机的 xl 设置，并将 `vnclisten` 更改为所需的值。请注意，以端口号 5902 为例，请仅指定 2，因为系统会自动加上 5900：

```
vfb = [ 'vnc=1,vnclisten="localhost:2"' ]
```

有关编辑 Guest 域的 xl 设置的详细信息，请参见第 23.1 节“XL — Xen 管理工具”。



提示

指派较大的端口号可避免与 VNC 查看器指派的端口号冲突，VNC 查看器使用的是最小的可用端口号。

25.3.2 使用 SDL 而不是 VNC 查看器

如果您要从虚拟机主机控制台访问虚拟机的显示画面（称为本地访问或机上访问），应使用 SDL 而不是 VNC 查看器。通过网络查看桌面时，VNC 查看器速度更快，但从同一台计算机查看桌面时，SDL 速度更快。

要设置为默认使用 SDL 而不是 VNC，请按如下所示更改虚拟机的配置信息。有关说明，请参阅第 23.1 节“[XL — Xen 管理工具](#)”。

```
vfb = [ 'sdl=1' ]
```

请记住，与 VNC 查看器窗口不同，关闭 SDL 窗口会终止虚拟机。

25.4 虚拟键盘

启动虚拟机后，主机会根据虚拟机的设置创建一个与 **keymap** 项匹配的虚拟键盘。如果未指定 **keymap** 项，虚拟机的键盘将默认为“英语（美国）”。

要查看虚拟机当前的 **keymap** 项，请在 Dom0 上输入以下命令：

```
tux > xl list -l VM_NAME | grep keymap
```

要配置 Guest 的虚拟键盘，请使用以下代码段：

```
vfb = [ 'keymap="de"' ]
```

有关支持的键盘布局的完整列表，请参见 [xl.cfg](#) 手册页 [man 5 xl.cfg](#) 的“[Keymaps](#)”部分。

25.5 分配专用 CPU 资源

在 Xen 中，可以指定 Dom0 或 VM Guest 应使用多少以及哪些 CPU 核心来保持其性能。Dom0 的性能对于系统整体而言非常重要，因为磁盘和网络驱动程序都是在 Dom0 上运行。此外，I/O 密集型 Guest 的工作负载可能会消耗 Dom0 的大量 CPU 周期。另一方面，设置 VM Guest 的目的是为了完成某项任务，而 VM Guest 的性能对于此类任务能否完成也很重要。

25.5.1 Dom0

为 Dom0 分配专用 CPU 资源可以提高虚拟化环境的总体性能，因为这样 Dom0 便会有可用的 CPU 时间处理来自 VM Guest 的 I/O 请求。不为 Dom0 分配专用 CPU 资源往往会导致性能不佳，并可能导致 VM Guest 无法正常运行。

分配专用 CPU 资源涉及到三个基本步骤：修改 Xen 引导行，将 Dom0 的 VCPU 绑定到物理处理器，然后在 VM Guest 上配置 CPU 相关选项：

1. 首先，您需要将 `dom0_max_vcpus=X` 追加到 Xen 引导行。为此，可将下面一行添加到 `/etc/default/grub`：

```
GRUB_CMDLINE_XEN="dom0_max_vcpus=X"
```

如果 `/etc/default/grub` 已包含设置了 `GRUB_CMDLINE_XEN` 的行，请将 `dom0_max_vcpus=X` 追加到此行。

需将 `X` 替换为供 Dom0 专用的 VCPU 数量。

2. 运行以下命令更新 GRUB 2 配置文件：

```
tux > sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. 重引导以使更改生效。

4. 下一步是将 Dom0 的每个 VCPU 绑定（或“固定”）到一个物理处理器。

```
tux > sudo xl vcpu-pin Domain-0 0 0  
xl vcpu-pin Domain-0 1 1
```

第一行将 Dom0 的 VCPU 0 绑定到物理处理器 0，第二行将 Dom0 的 VCPU 1 绑定到物理处理器 1。

5. 最后，需确保没有任何 VM Guest 使用 Dom0 的 VCPU 专用的物理处理器。假设您正在运行一个 8 CPU 系统，需将

```
cpus="2-8"
```

添加到相关 VM Guest 的配置文件。

25.5.2 VM Guest

用户常常需要分配特定的 CPU 资源供虚拟机专用。默认情况下，虚拟机会使用任何可用的 CPU 核心。为虚拟机指派合理数量的物理处理器可以提升其性能，因为指派给虚拟机后，其他 VM Guest 都不能使用这些物理处理器。假设某台计算机具有 8 个 CPU 核心，而虚拟机需要其中的 2 个，请按如下所示更改其配置文件：

```
vcpus=2  
cpus="2,3"
```

上面的示例为 VM Guest 分配了 2 个专用处理器，即第 3 个和第 4 个处理器（从 0 开始算起为 2 和 3）。如果您需要指派更多的物理处理器，请使用 `cpus="2-8"` 语法。

如果您需要以热插拔方式为名为 “alice” 的 Guest 更改 CPU 指派，请在相关 Dom0 上使用以下命令：

```
tux > sudo xl vcpu-set alice 2  
tux > sudo xl vcpu-pin alice 0 2  
tux > sudo xl vcpu-pin alice 1 3
```

此示例为该 Guest 分配了 2 个专用物理处理器，并将其 VCPU 0 和 VCPU 1 分别绑定到物理处理器 2 和 3。现在检查指派：

```
tux > 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

25.6 HVM 功能

在 Xen 中，某些功能仅适用于全虚拟化域。这些功能不常用到，但在某些环境中它们仍可能具有独特的作用。

25.6.1 指定引导时使用的引导设备

与使用物理硬件时一样，有时需要从其他设备而非 VM Guest 自己的引导设备来引导该 Guest。对于全虚拟计算机，可以在域 xl 配置文件中使用 `boot` 参数来选择引导设备：

```
boot = BOOT_DEVICE
```

`BOOT_DEVICE` 可以是 `c`（表示硬盘）、`d`（表示 CD-ROM）或 `n`（表示网络/PXE）。您可以指定多个选项，系统会按指定的顺序尝试这些选项。例如，

```
boot = dc
```

从 CD-ROM 引导，如果 CD-ROM 不可引导，则回退到硬盘。

25.6.2 更改 Guest 的 CPUID

要将 VM Guest 从一台 VM 主机服务器迁移到另一台 VM 主机服务器，VM Guest 系统只能使用这两个 VM 主机服务器系统上均会提供的 CPU 功能。如果两台主机上的实际 CPU 不同，可能需要在启动 VM Guest 之前隐藏某些功能。这样便可以在两台主机之间迁移 VM Guest。对于全虚拟化 Guest，可以通过配置可供 Guest 使用的 `cpuid` 来实现此目的。

要大致了解当前 CPU 的情况，请查看 `/proc/cpuinfo`。其中包含定义当前 CPU 的所有重要信息。

要重新定义 CPU，请先查看 CPU 供应商的相应 `cpuid` 定义。可从以下网站获取这些定义：

Intel

<http://www.intel.com/Assets/PDF/apnote/241618.pdf> 

```
cpuid = "host,tm=0,sse3=0"
```

语法为“键=值”对的逗号分隔列表，前面加上“host”一词。一些键接受数字值，其他所有键接受描述功能位作用的单个字符。有关 cpuid 键的完整列表，请参见 [man 5 xl.cfg](#)。可使用下面的值更改相应的位：

1

将对应的位强制设为 1

0

将对应的位强制设为 0

x

使用默认策略的值

k

使用主机定义的值

s

与 k 类似，但会在迁移后保留值

请注意，位是从位 0 开始从右向左计数的。

25.6.3 增加 PCI-IRQ 的数量

如果您需要增加 Dom0 和/或 VM Guest 可用的 PCI-IRQ 的默认数量，可以修改 Xen 内核命令行。使用 **extra_guest_irqs=** DOMU_IRGS,DOM0_IRGS 命令。第一个可选数字 DOMU_IRGS 是所有 VM Guest 公用的数字，第二个可选数字 DOM0_IRGS（前面带有逗号）用于 Dom0。更改 VM Guest 的设置不会影响 Dom0，反之亦然。例如，要在不更改 VM Guest 的情况下更改 Dom0，请使用

```
extra_guest_irqs=,512
```

25.7 虚拟 CPU 调度

Xen 超级管理程序会将虚拟 CPU 单独调度到不同的物理 CPU 中。在每个核心均支持多个线程的新式 CPU 中，虚拟 CPU 可以在同一核心上的不同线程中运行，因此会相互影响。在一个线程中运行的虚拟 CPU 的性能可能会受到其他线程中的其他虚拟 CPU 所执行操作的显著影响。如果这些虚拟 CPU 属于不同的 Guest 系统，这些 Guest 可能会相互影响。具体影响各不相同，从 Guest CPU 用时统计出现差异，到遭遇边信道攻击等更坏的情况。

调度粒度可以解决此问题。您可以使用 Xen 引导参数在引导时指定粒度：

```
sched-gran=GRANULARITY
```

请将 GRANULARITY 替换为以下值之一：

cpu

Xen 超级管理程序的常规调度。不同 Guest 的虚拟 CPU 可以共享同一个物理 CPU 核心。此为默认设置。

core

一个虚拟核心的虚拟 CPU 始终一起调度到一个物理核心中。来自不同虚拟核心的两个或更多个虚拟 CPU 永远不会调度到同一个物理核心中。因此，在某些物理核心中，可能有一部分 CPU 保持空闲状态，即使存在需要运行的虚拟 CPU 也不例外。对性能的影响取决于 Guest 系统内部正在运行的实际工作负载。在我们所分析的大多数案例中，观察到的性能下降情况（尤其是在承受很高负载的情况下）比禁用超线程（使所有核心仅保留一个线程）要轻一些（请参见 <https://xenbits.xen.org/docs/unstable/misc/xen-command-line.html#smt-x86> 上的 smt 引导选项）。

socket

粒度甚至可以提高到 CPU 插槽级别。

26 管理任务

26.1 引导加载程序

引导加载程序控制虚拟化软件的引导和运行方式。您可以使用 YaST 或者通过直接编辑引导加载程序配置文件来修改引导加载程序属性。

您可以单击 YaST > 系统 > 引导加载程序访问 YaST 引导加载程序。单击引导加载程序选项选项卡，然后选择包含 Xen 内核的行作为默认引导部分。



引导加载器设置

引导代码选项(D) 内核参数(K) 引导加载程序选项(L)

超时 (以秒计) (T) ☒ 探测外来操作系统(B) ☐ 引导时隐藏菜单(H)

默认引导项(D)

☐ 使用密码保护引导加载器(E) ☒ 仅保护启动项不被修改(R)

GRUB2 用户 root 的口令(P) 再次键入密码(T)

帮助(H) 取消(C) 确定(O)

图 26.1：引导加载程序设置

单击确定进行确认。您下次引导主机时，引导加载程序将准备好提供 Xen 虚拟化环境。

您可以使用引导加载程序来指定功能，例如：

- 传递内核命令行参数。
- 指定内核映像和初始 RAM 磁盘。

- 选择特定的超级管理程序。
- 向超级管理程序传递其他参数。有关完整的参数列表，请参见 <http://xenbits.xen.org/docs/unstable/misc/xen-command-line.html>。

可以通过编辑 `/etc/default/grub` 文件来自定义虚拟化环境。将下面一行添加到此文件：`GRUB_CMDLINE_XEN="<boot_parameters>"`。编辑该文件后，请不要忘记运行 `grub2-mkconfig -o /boot/grub2/grub.cfg`。

26.2 稀疏映像文件和磁盘空间

如果主机的物理磁盘已没有可用空间，使用基于稀疏映像文件的虚拟磁盘的虚拟机将无法向其磁盘写入数据。因此，它会报告 I/O 错误。

如果发生这种情况，您应该释放物理磁盘上的可用空间，重新装入虚拟机的文件系统，然后将文件系统重新设置为读写模式。

要检查稀疏映像文件的实际磁盘要求，请使用 `du -h <image file>` 命令。

要增加稀疏映像文件的可用空间，请先增加文件大小，然后增加文件系统的大小。



警告：在调整大小之前备份

改动分区或稀疏文件的大小始终要承担数据失败的风险。在未备份的情况下，请不要执行此操作。

可以在 VM Guest 运行时联机调整映像文件的大小。使用以下命令增加稀疏映像文件的大小：

```
tux > sudo dd if=/dev/zero of=<image file> count=0 bs=1M seek=<new size in MB>
```

例如，要将 `/var/lib/xen/images/sles/disk0` 文件的大小增至 16 GB，请使用以下命令：

```
tux > sudo dd if=/dev/zero of=/var/lib/xen/images/sles/disk0 count=0 bs=1M  
seek=16000
```




注意：增加非稀疏映像的大小

还可以增加设备的非稀疏映像文件的大小，但您必须知道上一个映像的具体结束位置。使用 `seek` 参数指向映像文件的末尾，然后使用如下所示的命令：

```
tux > sudo dd if=/dev/zero of=/var/lib/xen/images/sles/disk0 seek=8000  
bs=1M count=2000
```

请务必正确使用 `seek`，否则可能发生数据丢失情况。

如果在 VM Guest 运行时执行大小调整操作，另外还需调整向 VM Guest 提供映像文件的循环设备的大小。首先使用以下命令检测正确的循环设备：

```
tux > sudo losetup -j /var/lib/xen/images/sles/disk0
```

然后使用以下命令调整循环设备（例如 `/dev/loop0`）的大小：

```
tux > sudo losetup -c /dev/loop0
```

最后使用 `fdisk -l /dev/xvdb` 命令检查 Guest 系统中块设备的大小。设备名取决于实际增大的设备。

调整稀疏文件中的文件系统大小所涉及到的工具取决于实际文件系统。《储存管理指南》中对此做了详细说明。

26.3 迁移 Xen VM Guest 系统

在 Xen 中，可将 VM Guest 系统从一台 VM 主机服务器迁移到另一台 VM 主机服务器，而且几乎不会造成服务中断。例如，可以使用此功能将一个繁忙的 VM Guest 转移到一台硬件更强大或者尚无负载的 VM 主机服务器。或者，如果 VM 主机服务器的某项服务不能中断，可将此计算机上运行的所有 VM Guest 系统迁移到其他计算机，以避免该服务中断。这只是其中的两个例子 — 在您的个人使用情形中，可能还有许多其他迁移原因。

在开始之前，需要了解有关 VM 主机服务器的一些应预先注意的事项：

- 所有 VM 主机服务器系统应使用类似的 CPU。CPU 的频率并不是很重要，但它们应该使用同一 CPU 系列。要获取有关所用 CPU 的详细信息，请使用 `cat /proc/cpuinfo`。第 26.3.1 节“检测 CPU 功能”中提供了有关比较主机 CPU 功能的更多细节。
- 特定 Guest 系统使用的所有资源必须已在所有相关 VM 主机服务器系统上提供 — 例如，使用的所有块设备必须在两个 VM 主机服务器系统上均存在。
- 如果迁移过程涉及的主机在不同的子网中运行，请确保 Guest 可以使用 DHCP 中继，或者针对采用静态网络配置的 Guest 手动设置网络。
- 使用 `PCI 直通` 等特殊功能可能会造成问题。为可能会在不同 VM 主机服务器系统之间迁移 VM Guest 系统的环境部署虚拟机时，请不要实施这些功能。
- 为了快速迁移，必须设置一个快速网络。如果可能，请使用 GB 以太网和快速交换机。部署 VLAN 也可能有助于避免冲突。

26.3.1 检测 CPU 功能

可以使用 `cpuid` 和 `xen_maskcalc.py` 工具，将您要从中迁移源 VM Guest 的主机上的 CPU 功能与目标主机上的 CPU 功能进行比较。这样，您就可以更好地预测 Guest 迁移是否会成功。

1. 在预期要运行或接收迁移的 VM Guest 的每个 Dom0 上运行 `cpuid -lr` 命令，然后在文本文件中捕获输出，例如：

```
tux@vm_host1 > sudo cpuid -lr > vm_host1.txt
tux@vm_host2 > sudo cpuid -lr > vm_host2.txt
tux@vm_host3 > sudo cpuid -lr > vm_host3.txt
```

2. 将所有输出文本文件复制到装有 `xen_maskcalc.py` 脚本的主机上。
3. 针对所有输出文本文件运行 `xen_maskcalc.py` 脚本：

```
tux > sudo xen_maskcalc.py vm_host1.txt vm_host2.txt vm_host3.txt
cpuid = [
    "0x00000001:ecx=x00xxxxxx0xxxxxxxx00xxxxxxxx",
    "0x00000007,0x00:ebx=xxxxxxxxxxxxxxxx00x0000x0x0x00"
```

```
]

```

4. 将输出的 `cpuid=[...]` 配置片段复制到迁移的 Guest `domU.cfg` 的 `xl` 配置中，或复制到其 `libvirt` 的 XML 配置中。
5. 使用经过删减的 CPU 配置启动源 Guest。现在，Guest 只能使用每台主机上提供的 CPU 功能。



提示

`libvirt` 还支持计算用于迁移的基线 CPU。有关详细信息，请参考 <https://documentation.suse.com/sles/15-SP2/html/SLES-all/article-vt-best-practices.html>。

26.3.1.1 更多信息

<http://etallen.com/cpuid.html> 上提供了有关 `cpuid` 的更多细节。

您可以从 https://github.com/twizted/xen_maskcalc 下载最新版本的 CPU 掩码计算器。

26.3.2 准备要迁移的块设备

VM Guest 系统所需的块设备必须已在所有相关 VM 主机服务器系统上提供。要做到这一点，可以实施某种共享储存来充当所迁移 VM Guest 系统的根文件系统的容器。常见的做法包括：

- 可以设置 `iSCSI`，以便可从不同的系统同时访问相同的块设备。有关 `iSCSI` 的详细信息，请参见《储存管理指南》，第 14 章“经由 IP 网络的大容量储存：iSCSI”。
- `NFS` 是广泛使用的根文件系统，用户可从不同的位置轻松访问该文件系统。有关更多信息，请参见《管理指南》，第 33 章“通过 NFS 共享文件系统”。
- 如果只涉及到两个 VM 主机服务器系统，则可以使用 `DRBD`。这会在一定程度上提高数据安全性，因为此方法会通过网络镜像使用的数据。有关详细信息，请参见 <https://documentation.suse.com/sle-ha/15-SP2/> 上的 SUSE Linux Enterprise High Availability Extension 15 SP2 文档。

- 如果提供的硬件允许对相同磁盘进行共享访问，则还可以使用 [SCSI](#)。
- [NPIV](#) 是使用光纤通道磁盘的特殊模式。但在这种情况下，所有迁移主机都必须挂接到同一个光纤通道交换机。有关 NPIV 的详细信息，请参见第 24.1 节 “[将物理储存映射到虚拟磁盘](#)”。一般而言，如果光纤通道环境支持 4 Gbit 或更快的连接，便可使用此方式。

26.3.3 迁移 VM Guest 系统

VM Guest 系统的实际迁移是使用以下命令完成的：

```
tux > sudo xl migrate <domain_name> <host>
```

迁移速度取决于将占用内存的内容保存到磁盘、发送到 VM 主机服务器并在该处装载的速度。也就是说，迁移小型 VM Guest 系统可能比迁移包含大量内存的大型系统的速度更快。

26.4 监视 Xen

在对众多虚拟 Guest 进行常规运维时，检查所有不同 VM Guest 系统的健全性是必不可少的功能。除了系统工具以外，Xen 还提供了数个工具用于收集有关系统的信息。



提示：监视 VM 主机服务器

可以通过虚拟机管理器监视 VM 主机服务器的基本情况（I/O 和 CPU）。有关详细信息，请参考第 10.8.1 节 “[使用虚拟机管理器进行监视](#)”。

26.4.1 使用 [xentop](#) 监视 Xen

用于收集有关 Xen 虚拟环境的信息的首选终端应用程序是 [xentop](#)。遗憾的是，此工具需要一个相当宽的终端，否则它会在显示内容中插入换行符。

[xentop](#) 提供多个命令键，可供您查看有关所监视系统的详细信息。其中较为重要的键包括：

D

更改屏幕两次刷新相隔的延迟时间。

N

同时显示网络统计信息。请注意，显示的只有标准配置。如果您使用路由网络等特殊配置，将不会显示网络。

B

显示相应的块设备及其累计使用次数。

有关 **xentop** 的详细信息，请参见手册页 **man 1 xentop**。



提示：virt-top

libvirt 提供不限定超级管理程序的工具 **virt-top**，建议使用此工具来监视 VM Guest。有关详细信息，请参见第 10.8.2 节“使用 **virt-top** 进行监视”。

26.4.2 其他工具

我们还提供了许多系统工具来帮助您监视或调试运行中的 SUSE Linux Enterprise 系统。《系统分析和微调指南》，第 2 章“系统监视实用程序”中介绍了其中一些工具。以下工具特别适合用于监视虚拟化环境：

ip

命令行实用程序 **ip** 可用于监视任意网络接口。如果您设置了路由网络或者应用了掩蔽网络，此实用程序特别有用。要监视名为 **alice.0** 的网络接口，请运行以下命令：

```
tux > watch ip -s link show alice.0
```

bridge

在标准设置中，所有 Xen VM Guest 系统都会挂接到虚拟网桥。使用 **bridge** 可以确定 VM Guest 系统中网桥与虚拟网络适配器之间的连接。例如，**bridge link** 的输出可能如下所示：

```
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
```

此输出表明系统上定义了两个虚拟网桥。一个网桥连接到物理以太网设备 `eth0`，另一个网桥连接到 VLAN 接口 `vnet0`。

iptables-save

使用掩蔽网络时，或者设置了多个以太网接口并与防火墙设置结合使用时，此工具特别有用，它可以帮助检查当前的防火墙规则。

iptables 命令可用于检查所有不同的防火墙设置。要列出某个链甚至整个设置的所有规则，可以使用 **iptables-save** 或 **iptables -S** 命令。

26.5 提供 VM Guest 系统的主机信息

在标准 Xen 环境中，VM Guest 系统只能获得有关运行它的 VM 主机服务器系统的极为有限的信息。如果某个 Guest 应该了解有关运行它的 VM 主机服务器的更多信息，`vhostmd` 可为选定 Guest 提供详细信息。要设置系统以运行 `vhostmd`，请执行以下操作：

1. 在 VM 主机服务器上安装 `vhostmd` 软件包。
2. 要在配置中添加或删除 `metric` 部分，请编辑文件 `/etc/vhostmd/vhostmd.conf`。不过，默认设置也可正常工作。
3. 使用以下命令检查 `vhostmd.conf` 配置文件的有效性：

```
tux > cd /etc/vhostmd
tux > xmllint --postvalid --noout vhostmd.conf
```

4. 使用 **`sudo systemctl start vhostmd`** 命令启动 `vhostmd` 守护程序。
如果系统启动期间应自动启动 `vhostmd`，请运行以下命令：

```
tux > sudo systemctl enable vhostmd
```

5. 使用以下命令将映像文件 `/dev/shm/vhostmd0` 挂接到名为 `alice` 的 VM Guest 系统：

```
tux > xl block-attach opensuse /dev/shm/vhostmd0,,xvdb,ro
```

6. 在 VM Guest 系统上登录。
7. 安装客户端软件包 `vm-dump-metrics`。

8. 运行 **`vm-dump-metrics`** 命令。要将结果保存到文件中，请使用选项 **`-d`** **`<filename>`**。

`vm-dump-metrics` 返回的结果为 XML 输出。相应的度量项遵循 DTD **`/etc/vhostmd/metric.dtd`**。

有关详细信息，请参见 VM 主机服务器系统上的手册页 **`man 8 vhostmd`** 和 **`/usr/share/doc/vhostmd/README`**。在 Guest 上，请参见手册页 **`man 1 vm-dump-metrics`**。

27 XenStore：在域之间共享的配置数据库

本章介绍有关 XenStore 的基本信息、其在 Xen 环境中的作用、XenStore 所使用的文件的目录结构，以及 XenStore 命令的说明。

27.1 简介

XenStore 是在 Dom0 中所运行的 VM Guest 与管理工具之间共享的配置和状态信息数据库。VM Guest 和管理工具对 XenStore 进行读取和写入以传达配置信息、状态更新和状态更改。XenStore 数据库由 Dom0 管理，支持读取和写入密钥等简单操作。可以通过监测关注项将 XenStore 中发生的任何更改通知给 VM Guest 和管理工具。请注意，`xenstored` 守护程序由 `xencommons` 服务管理。

XenStore 位于 Dom0 上的单个数据库文件 `/var/lib/xenstored/tdb`（`tdb` 表示树数据库）中。

27.2 文件系统接口

XenStore 数据库内容由与 `/proc` 类似的虚拟文件系统表示（有关 `/proc` 的详细信息，请参见《系统分析和微调指南》，第 2 章“系统监视实用程序”，第 2.6 节“`/proc` 文件系统”）。树有三条主路径：`/vm`、`/local/domain` 和 `/tool`。

- `/vm` - 储存有关 VM Guest 配置的信息。
- `/local/domain` - 储存有关本地节点上的 VM Guest 的信息。
- `/tool` - 储存有关各种工具的一般信息。



提示

每个 VM Guest 都有两个不同的 ID 编号。即使 VM Guest 迁移到其他计算机，全局唯一标识符 (UUID) 也保持不变。域标识符 (DOMID) 是表示正在运行的特定实例的标识号。将 VM Guest 迁移到其他计算机后，此编号通常会改变。

27.2.1 XenStore 命令

可使用以下命令操作 XenStore 数据库的文件系统结构：

xenstore-ls

显示 XenStore 数据库的完整转储。

xenstore-read path_to_xenstore_entry

显示指定 XenStore 项的值。

xenstore-exists xenstore_path

报告指定的 XenStore 路径是否存在。

xenstore-list xenstore_path

显示指定 XenStore 路径的所有子项。

xenstore-write path_to_xenstore_entry

更新指定 XenStore 项的值。

xenstore-rm xenstore_path

去除指定的 XenStore 项或目录。

xenstore-chmod xenstore_path mode

更新对指定 XenStore 路径的读取/写入权限。

xenstore-control

将一条命令（例如，用于触发完整性检查）发送到 xenstored 后端。

27.2.2 /vm

/vm 路径按每个 VM Guest 的 UUID 编制索引，用于储存虚拟 CPU 数量和分配的内存量等配置信息。每个 VM Guest 都有一个 /vm/<uuid> 目录。要列出目录内容，请使用 **xenstore-list**。

```
tux > sudo xenstore-list /vm
000000000-0000-0000-0000-000000000000
9b30841b-43bc-2af9-2ed3-5a649f466d79-1
```

输出的第一行属于 Dom0，第二行属于正在运行的 VM Guest。以下命令会列出与 VM Guest 相关的所有项：

```
tux > 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
```

要读取某个项（例如，专用于 VM Guest 的虚拟 CPU 数量）的值，请使用 **xenstore-read**：

```
tux > sudo xenstore-read /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1/vcpus
1
```

选定 /vm/<uuid> 项的列表如下：

uuid

VM Guest 的 UUID，在迁移过程中不会变化。

on_reboot

指定响应重引导请求时是要销毁还是重启动 VM Guest。

on_poweroff

指定响应暂停请求时是要销毁还是重启动 VM Guest。

on_crash

指定响应崩溃事件时是要销毁还是重启动 VM Guest。

vcpus

分配给 VM Guest 的虚拟 CPU 数量。

vcpu_avail

VM Guest 的活动虚拟 CPU 的位掩码。位掩码中有多个位等于 vcpus 的值，其中有一个位是为每个联机虚拟 CPU 设置的。

name

VM Guest 的名称。

普通的 VM Guest（不是 Dom0）使用 /vm/<uuid>/image 路径：

```
tux > sudo xenstore-list /vm/9b30841b-43bc-2af9-2ed3-5a649f466d79-1/image
ostype
kernel
cmdline
ramdisk
dmargs
device-model
display
```

使用的项的解释如下：

ostype

VM Guest 的操作系统类型。

kernel

Dom0 上指向 VM Guest 内核的路径。

cmdline

引导时对 VM Guest 使用的内核命令行。

ramdisk

Dom0 上指向 VM Guest RAM 磁盘的路径。

dmargs

显示传递给 QEMU 进程的参数。如果您使用 ps 查看 QEMU 进程，看到的参数应该与 /vm/<uuid>/image/dmargs 中的相同。

27.2.3 `/local/domain/<domid>`

此路径按运行中的域 (VM Guest) ID 编制索引，包含有关运行中的 VM Guest 的信息。请记住，在迁移 VM Guest 期间，域 ID 会变化。可用的项如下：

vm

此 VM Guest 的 `/vm` 目录的路径。

on_reboot、on_poweroff、on_crash、name

请参见第 27.2.2 节 “`/vm`” 中的相同选项。

domid

VM Guest 的域标识符。

cpu

当前 VM Guest 固定到的 CPU。

cpu_weight

出于调度目的指派给 VM Guest 的权重。权重越高，使用物理 CPU 的频率就越高。

除了上面所述的各个项以外，`/local/domain/<domid>` 下的多个子目录也包含特定的项。要查看所有可用的项，请参见《XenStore Reference》（XenStore 参考）(http://wiki.xen.org/wiki/XenStore_Reference)。

/local/domain/<domid>/memory

包含内存信息。`/local/domain/<domid>/memory/target` 包含 VM Guest 的目标内存大小（以 KB 为单位）。

/local/domain/<domid>/console

包含有关 VM Guest 所用控制台的信息。

/local/domain/<domid>/backend

包含有关 VM Guest 所用的所有后端设备的信息。该路径包含 VM Guest 自己的子目录。

/local/domain/<domid>/device

包含有关 VM Guest 的前端设备的信息。

/local/domain/<domid>/device-misc

包含有关设备的其他信息。

/local/domain/<domid>/store

包含有关 VM Guest 的储存的信息。

28 使用 Xen 作为高可用性虚拟化主机

与在专用硬件上运行每台服务器的设置相比，将两台 Xen 主机设置为故障转移系统可以带来多项优势。

- 一台服务器发生故障不会导致出现重大服务中断情况。
- 购置一台大型计算机通常比购置多台小型计算机要便宜。
- 按需添加新服务器的工作简单无比。
- 服务器的利用率可得到改进，而这又会对系统的电源消耗产生正面影响。

第 26.3 节“迁移 Xen VM Guest 系统”中介绍了 Xen 主机的迁移设置。下面描述了几种典型方案。

28.1 使用远程储存实现 Xen HA

Xen 可以直接向相应的 Xen Guest 系统提供多个远程块设备。这些设备包括 iSCSI、NPIV 和 NBD。所有这些设备都可用于执行实时迁移。储存系统准备就绪后，请先尝试使用您已在网络中使用的相同设备类型。

如果储存系统不可直接使用，但具有通过 NFS 提供所需空间的能力，您也可以在 NFS 上创建映像文件。如果所有 Xen 主机系统上都提供了 NFS 文件系统，则此方法还可以实现 Xen Guest 的实时迁移。

设置新系统时，其中一个主要考虑因素为是否应该实施专用的储存区域网络。可行的方法如下：

表 28.1：XEN 远程储存

方法	复杂性	注释
以太网	低	请注意，所有块设备流量都将通过用于传送网络流量的同一以太网接口传送。这可能会使 Guest 的性能受到限制。

方法	复杂性	注释
专用于储存的以太网。	中	通过专用以太网接口运行储存流量可以消除服务器端的瓶颈。但是，为自己的网络规划自己的 IP 地址范围时，以及规划专用于储存的 VLAN（可能有此需要）时，您需要考虑很多因素。
NPIV	高	NPIV 是用于虚拟化光纤通道连接的一种方法。通过使用最低支持 4 Gbit/秒的数据传输速率并允许复杂储存系统设置的适配器来提供此功能。

通常，1 Gbit/秒的以太网设备可以充分利用典型的硬盘或储存系统。在使用极快的储存系统时，此类以太网设备也许会限制系统的速度。

28.2 使用本地储存实现 Xen HA

出于空间和预算原因，有时可能需要依赖于 Xen 主机系统本地的储存。如果仍想实现实时迁移，需要构建镜像到两台 Xen 主机的块设备。可用于执行此操作的软件称为分布式复制块设备 (DRBD)。

如果需要设置一个使用 DRBD 在两台 Xen 主机之间镜像块设备或文件的系统，这两台主机应使用相同的硬件。如果其中一台主机的硬盘速度较慢，则两台主机的速度会受到同样的限制。

在设置期间，所需的每个块设备均应使用自己的 DRBD 设备。设置此类系统是一个相当复杂的任务。

28.3 Xen HA 和专用网桥

使用需要相互通讯的多个 Guest 系统时，可以通过普通的接口来进行通讯。但出于安全原因，建议您创建一个仅连接到 Guest 系统的网桥。

在还需要支持实时迁移的 HA 环境中，此类专用网桥必须连接到其他 Xen 主机。使用专用物理以太网设备和专用网络可以做到这一点。

另一种实现方法是使用 VLAN 接口。在这种情况下，所有流量都将通过普通以太网接口传送。不过，VLAN 接口不会收到普通流量，因为只有标记为要传送到正确 VLAN 的 VLAN 包会被转发。

有关 VLAN 接口设置的详细信息，请参见第 13.1.3 节 “使用 VLAN 接口”。

V 使用 QEMU 管理虚拟机

- 29 QEMU 概述 258
- 30 设置 KVM VM 主机服务器 259
- 31 Guest 安装 270
- 32 使用 qemu-system-ARCH 运行虚拟机 286
- 33 使用 QEMU 监视器管理虚拟机 314

29 QEMU 概述

QEMU 是快速的跨平台开源计算机模拟器，可为您模拟数量庞大的硬件体系结构。QEMU 可让您在现有系统（VM 主机服务器）之上运行未经修改的完整操作系统 (VM Guest)。

您还可以使用 QEMU 进行调试 — 可以轻松停止运行中的虚拟机、检查其状态、保存并在以后恢复其状态。

QEMU 由以下部分构成：

- 处理器模拟器（x86、IBM Z、PowerPC、Sparc）
- 模拟的设备（显卡、网卡、硬盘、鼠标）
- 用于将被模拟设备连接到相关主机设备的通用设备
- 被模拟计算机（PC、Power Mac）的说明
- 调试程序
- 用来与模拟器交互的用户界面

QEMU 是 KVM 和 Xen 虚拟化的核心，在这些虚拟化环境中提供常规的计算机模拟。Xen 在使用 QEMU 时会对用户隐藏部分功能，而 KVM 在使用 QEMU 时会透明地公开大部分 QEMU 功能。如果 VM Guest 硬件体系结构与 VM 主机服务器的体系结构相同，QEMU 便可以利用 KVM 加速的优势（SUSE 仅支持装载了 KVM 加速的 QEMU）。

除了提供核心虚拟化基础结构以及特定于处理器的驱动程序以外，QEMU 还提供特定于体系结构的用户空间程序来管理 VM Guest。根据具体的体系结构，此程序是以下其中一项：

- `qemu-system-i386`
- `qemu-system-s390x`
- `qemu-system-x86_64`

在后面的章节中将此命令称为 `qemu-system-ARCH`；而示例中使用的则是 `qemu-system-x86_64` 命令。

30 设置 KVM VM 主机服务器

本节介绍如何设置并使用 SUSE Linux Enterprise Server 15 SP2 作为基于 QEMU-KVM 的虚拟机主机。



提示：资源

一般情况下，虚拟 Guest 系统所需的硬件资源与将其安装在物理机上时所需的资源相同。您打算在主机系统上运行的 Guest 越多，需要添加到 VM 主机服务器的硬件资源（CPU、磁盘、内存和网络）就越多。

30.1 CPU 的虚拟化支持

要运行 KVM，您的 CPU 必须支持虚拟化，并且需要在 BIOS 中启用虚拟化。[/proc/cpuinfo](#) 文件包含有关 CPU 功能的信息。

要确定您的系统是否支持虚拟化，请参见[第 7.3 节 “KVM 硬件要求”](#)。

30.2 要求的软件

需在 KVM 主机上安装多个软件包。要安装所有必要的软件包，请执行以下操作：

1. 确认是否已安装 [yast2-vm](#) 软件包。此软件包是 YaST 的配置工具，可以简化虚拟化超级管理程序的安装过程。
2. 运行 YaST › 虚拟化 › 安装超级管理程序和工具。

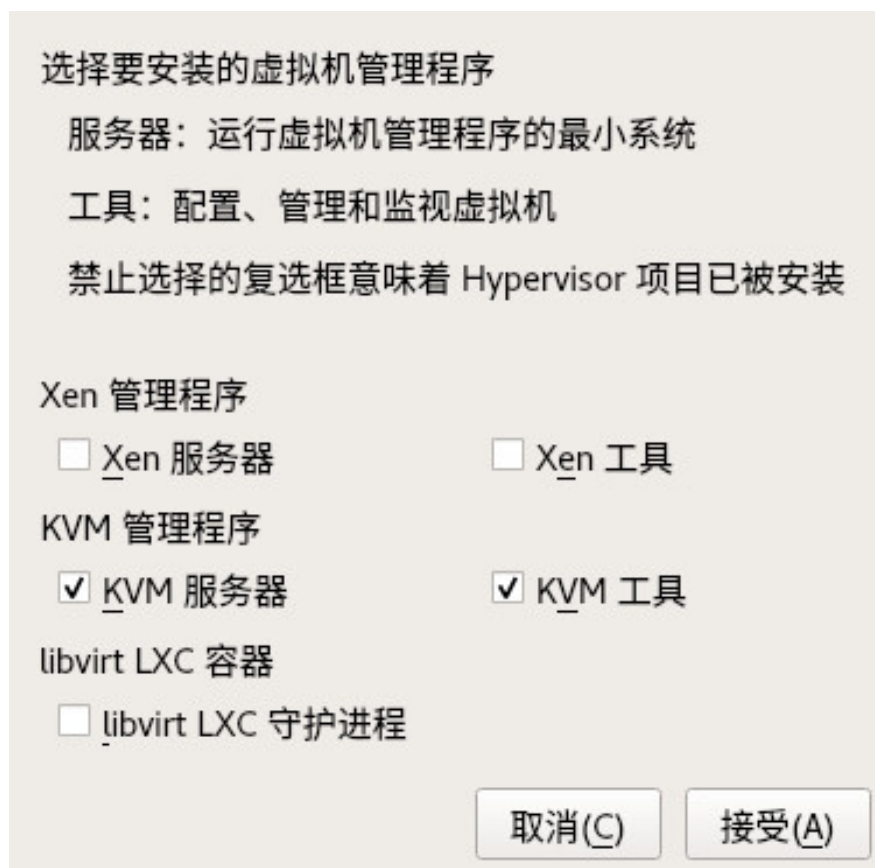


图 30.1：安装 KVM 超级管理程序和工具

3. 选择 KVM 服务器，最好也选择 KVM 工具，然后单击接受以确认。
4. 在安装过程中，您可以选择性地让 YaST 自动为您创建网桥。如果您不打算另外为虚拟 Guest 使用一块物理网卡，那么将 Guest 计算机连接到网络的标准方式就是使用网桥。

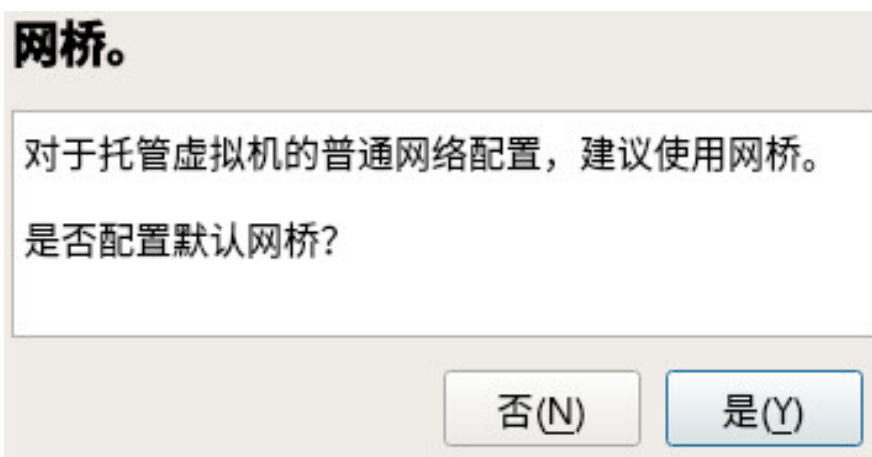


图 30.2：网桥

5. 安装所有所需的软件包（并激活新网络设置）后，尝试装载适用于您的 CPU 类型的 KVM 内核模块 — `kvm-intel` 或 `kvm-amd`：

```
root # modprobe kvm-intel
```

检查该模块是否已装载到内存中：

```
tux > lsmod | grep kvm
kvm_intel          64835  6
kvm                411041  1 kvm_intel
```

现在，KVM 主机即可为 KVM VM Guest 提供服务。有关更多信息，请参见第 32 章“使用 `qemu-system-ARCH` 运行虚拟机”。

30.3 特定于 KVM 主机的功能

您可以让基于 KVM 的 VM Guest 充分使用 VM 主机服务器硬件的特定功能（半虚拟化），以此提高这些 Guest 的性能。本节将介绍可以通过哪些方法来使 Guest 直接访问物理主机的硬件（无需模拟层），以充分利用这些硬件。



提示

本节中的示例假设读者基本了解 `qemu-system-ARCH` 命令行选项。有关更多信息，请参见第 32 章 “使用 `qemu-system-ARCH` 运行虚拟机”。

30.3.1 使用具有 `virtio-scsi` 的主机储存

`virtio-scsi` 是 KVM 的高级储存堆栈。它取代了以前用于 SCSI 设备直通的 `virtio-blk` 堆栈。与 `virtio-blk` 相比，它具有多项优势：

提高了可缩放性

KVM Guest 的 PCI 控制器数量有限，导致可挂接的设备数量也受到限制。`virtio-scsi` 解决了这个限制，因为它可以将多个储存设备组合到单个控制器上。`virtio-scsi` 控制器上的每个设备以逻辑单元 (LUN) 表示。

标准命令集

`virtio-blk` 使用 `virtio-blk` 驱动程序和虚拟机监视器均需知道的一小组命令，因此引入新命令需要同时更新该驱动程序和监视器。

相比之下，`virtio-scsi` 并不定义命令，而是遵循行业标准 SCSI 规范为这些命令定义一个传输协议。此方法与光纤通道、ATAPI 和 USB 设备等其他技术共享。

设备命名

`virtio-blk` 设备在 Guest 中显示为 `/dev/vdX`，这与物理系统中的设备名称不同，可能导致迁移时出现问题。

`virtio-scsi` 会确保设备名称与物理系统上的名称相同，这样便可轻松重新定位虚拟机。

SCSI 设备直通

对于由主机上整个 LUN 提供支持的虚拟磁盘，最好让 Guest 直接向 LUN 发送 SCSI 命令（直通）。此功能在 `virtio-blk` 中受到限制，因为 Guest 需使用 `virtio-blk` 协议而不是 SCSI 命令直通，此外，此功能并不适用于 Windows Guest。`virtio-scsi` 原生消除了这些限制。

30.3.1.1 virtio-scsi 的用法

KVM 支持对 `virtio-scsi-pci` 设备使用 SCSI 直通功能：

```
root # qemu-system-x86_64 [...] \  
-device virtio-scsi-pci,id=scsi
```

30.3.2 使用 vhost-net 实现加速网络

`vhost-net` 模块用于加速 KVM 的半虚拟化网络驱动程序。它可提供更低的延迟和更高的网络吞吐量。通过以下示例命令行启动 Guest 即可使用 `vhost-net` 驱动程序：

```
root # 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
```

请注意，`guest0` 是 vhost 驱动的设备的标识字符串。

30.3.3 使用多队列 virtio-net 提升网络性能

QEMU 提供了使用多队列提升网络性能的方式，来应对 VM Guest 中的虚拟 CPU 数量增加的情况。多队列 virtio-net 允许 VM Guest 的虚拟 CPU 并行传输包，因此可以提升网络性能。VM 主机服务器和 VM Guest 端都需要提供多队列支持。



提示：性能优势

多队列 virtio-net 解决方案在以下情况下最有利：

- 网络流量包较大。
- VM Guest 上存在许多同时保持活动状态的连接，这些连接主要是在 Guest 系统之间、Guest 与主机之间，或者 Guest 与外部系统之间建立的。
- 活动队列数量等于 VM Guest 中虚拟 CPU 的数量。

注意

尽管多队列 virtio-net 可以增加总网络吞吐量，但由于使用了虚拟 CPU 的计算资源，因此也会增加 CPU 消耗量。

过程 30.1：如何启用多队列 VIRTIO-NET

下面的过程列出了使用 **qemu-system-ARCH** 启用多队列功能的重要步骤。假设在 VM 主机服务器上设置了一个具有多队列功能（自内核版本 3.8 开始支持此功能）的 tap 网络设备。

1. 在 **qemu-system-ARCH** 中，为该 tap 设备启用多队列：

```
-netdev tap,vhost=on,queues=2*N
```

其中 **N** 代表队列对的数量。

2. 在 **qemu-system-ARCH** 中，为 virtio-net-pci 设备启用多队列并指定 MSI-X（消息信号式中断）矢量。

```
-device virtio-net-pci,mq=on,vectors=2*N+2
```

在用于计算 MSI-X 矢量数量的公式中：**N** 个矢量用于 TX（传输）队列，**N** 个矢量用于 RX（接收）队列，一个矢量用于配置目的，一个矢量用于可能的 VQ（矢量量化）控制。

3. 在 VM Guest 中的相关网络接口（在本示例中为 **eth0**）上启用多队列：

```
tux > sudo ethtool -L eth0 combined 2*N
```

最终的 **qemu-system-ARCH** 命令行类似于以下示例：

```
qemu-system-x86_64 [...] -netdev tap,id=guest0,queues=8,vhost=on \  
-device virtio-net-pci,netdev=guest0,mq=on,vectors=10
```

请注意，对于命令行的两个选项，需指定相同的网络设备 **id** (**guest0**)。

在运行中的 VM Guest 内部，以 **root** 特权指定以下命令：

```
tux > sudo ethtool -L eth0 combined 8
```


现在，Guest 系统网络将使用 **qemu-system-ARCH** 超级管理程序的多队列支持。

30.3.4 VFIO：对设备进行安全的直接访问

将 PCI 设备直接指派到 VM Guest（PCI 直通）可以避免在性能关键型路径中进行任何模拟，从而避免性能问题。VFIO 取代了传统的 KVM PCI 直通设备指派。要使用此功能，VM 主机服务器配置必须符合**重要：VFIO 和 SR-IOV 的要求**中所述的要求。

要通过 VFIO 将 PCI 设备指派到 VM Guest，需要确定该设备属于哪个 IOMMU 组。**IOMMU**（用于将支持直接内存访问的 I/O 总线连接到主内存的输入/输出内存管理单元）API 支持组表示法。组是可与系统中的所有其他设备相互隔离的一组设备。因此，组是 **VFIO** 使用的所有权单元。

过程 30.2：通过 VFIO 将 PCI 设备指派到 VM GUEST

1. 标识要指派到 Guest 的主机 PCI 设备。

```
tux > sudo lspci -nn
[...]
00:10.0 Ethernet controller [0200]: Intel Corporation 82576 \
Virtual Function [8086:10ca] (rev 01)
[...]
```

记下设备 ID（在本例中为 00:10.0）和供应商 ID (8086:10ca)。

2. 确定此设备的 IOMMU 组：

```
tux > sudo readlink /sys/bus/pci/devices/0000\:00\:10.0/iommu_group
../../../../kernel/iommu_groups/20
```

此设备的 IOMMU 组为 20。现在，您可以检查该设备是否属于同一个 IOMMU 组：

```
tux > 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. 从设备驱动程序取消绑定设备：

```
tux > sudo echo "0000:01:10.0" > /sys/bus/pci/devices/0000\:01\:10.0/  
driver/unbind
```

4. 使用步骤 1 中记下的供应商 ID 将设备绑定到 vfio-pci 驱动程序：

```
tux > sudo echo "8086 153a" > /sys/bus/pci/drivers/vfio-pci/new_id
```

系统即会创建一个新设备 `/dev/vfio/IOMMU_GROUP`，在本例中为 `/dev/vfio/20`。

5. 更改新建设备的所有权：

```
tux > sudo chown qemu.qemu /dev/vfio/DEVICE
```

6. 现在，运行为其指派了 PCI 设备的 VM Guest。

```
tux > sudo qemu-system-ARCH [...] -device  
vfio-pci,host=00:10.0,id=ID
```

！ 重要：不支持热插拔

从 SUSE Linux Enterprise Server 15 SP2 开始，不支持通过 VFIO 热插拔传递给 VM Guest 的 PCI 设备。

`/usr/src/linux/Documentation/vfio.txt` 文件中提供了有关 **VFIO** 驱动程序的更详细信息（需要安装软件包 `kernel-source`）。

30.3.5 VirtFS：在主机与 Guest 之间共享目录

VM Guest 通常在单独的计算空间中运行 — 它们各自都有自己的内存范围、专用的 CPU 和文件系统空间。能够共享 VM 主机服务器文件系统的某些部分，就能通过简化相互数据交换来提高虚拟化环境的灵活性。网络文件系统（例如 CIFS 和 NFS）是共享目录的传统方式，但由于它们不是专为虚拟化目的而设计，因此会受制于重大的性能和功能问题。

KVM 引入了一种经过优化的新方法，称为 VirtFS（有时称为“文件系统直通”）。VirtFS 使用半虚拟文件系统驱动程序，可以避免将 Guest 应用程序文件系统操作转换为块设备操作，然后将块设备操作转换为主机文件系统操作。

VirtFS 通常可用于以下情况：

- 要从多个 Guest 访问共享目录，或者提供 Guest 到 Guest 的文件系统访问。
- 在 Guest 引导过程中，要将虚拟磁盘替换为 Guest 的 RAM 磁盘所要连接到的根文件系统。
- 要从云环境中的单个主机文件系统为不同的客户提供储存服务。

30.3.5.1 实施

在 QEMU 中，可以通过定义两种类型的服务来简化 VirtFS 的实现：

- 用于在主机与 Guest 之间传输协议消息和数据的 `virtio-9p-pci` 设备。
- 用于定义导出文件系统属性（例如文件系统类型和安全模型）的 `fsdev` 设备。

例 30.1：使用 VIRTFS 导出主机的文件系统

```
tux > sudo qemu-system-x86_64 [...] \  
-fsdev local,id=expl❶,path=/tmp/❷,security_model=mapped❸ \  
-device virtio-9p-pci,fsdev=expl❹,mount_tag=v_tmp❺
```

- ❶ 要导出的文件系统的标识。
- ❷ 要导出的主机上的文件系统路径。
- ❸ 要使用的安全模型 — `mapped` 可使 Guest 文件系统模式和权限与主机相互隔离，而 `none` 会调用“直通”安全模型，其中，对 Guest 文件进行的权限更改也会反映在主机上。
- ❹ 前面使用 `-fsdev id=` 定义的导出的文件系统 ID。
- ❺ 稍后要用在 Guest 上装入所导出文件系统的装入标记。

可按如下所示在 Guest 上装入此类导出的文件系统：

```
tux > sudo mount -t 9p -o trans=virtio v_tmp /mnt
```

其中，`v_tmp` 是前面使用 `-device mount_tag=` 定义的装入标记，`/mnt` 是要将导出的文件系统装入到的安装点。

30.3.6 KSM：在 Guest 之间共享内存页

内核同页合并 (KSM) 是 Linux 内核的一项功能，可将多个运行中进程的相同内存页合并到一个内存区域中。由于 KVM Guest 在 Linux 中以进程的形式运行，KSM 为超级管理程序提供了内存过量使用功能，以提高内存的使用效率。因此，如果您需要在内存有限的主机上运行多个虚拟机，KSM 可能有所帮助。

KSM 将其状态信息储存在 `/sys/kernel/mm/ksm` 目录下的文件中：

```
tux > ls -l /sys/kernel/mm/ksm
full_scans
merge_across_nodes
pages_shared
pages_sharing
pages_to_scan
pages_unshared
pages_volatile
run
sleep_millisecs
```

有关 `/sys/kernel/mm/ksm/*` 文件含义的详细信息，请参见 `/usr/src/linux/Documentation/vm/ksm.txt`（软件包 `kernel-source`）。

要使用 KSM，请执行以下操作。

1. 尽管 SLES 在内核中包含了 KSM 支持，但其默认处于禁用状态。要启用该支持，请运行以下命令：

```
root # echo 1 > /sys/kernel/mm/ksm/run
```

2. 现在，请在 KVM 中运行多个 VM Guest，并检查 `pages_sharing` 和 `pages_shared` 文件的内容，例如：

```
tux > while [ 1 ]; do cat /sys/kernel/mm/ksm/pages_shared; sleep 1; done
```

13522

13523

13519

13518

13520

13520

13528

31 Guest 安装

`virt-manager` 和 `virt-install` 等基于 `libvirt` 的工具提供了方便的界面用于设置和管理虚拟机。这些工具充当 `qemu-system-ARCH` 命令的某种封装程序。不过，您也可以直接使用 `qemu-system-ARCH`，而无需使用基于 `libvirt` 的工具。



警告：qemu-system-ARCH 和 libvirt

使用 `qemu-system-ARCH` 创建的虚拟机对于基于 `libvirt` 的工具不可见。

31.1 使用 qemu-system-ARCH 进行基本安装

在下面的示例中，将为 SUSE Linux Enterprise Server 11 安装创建一个虚拟机。有关命令的详细信息，请参见相关的手册页。

如果您尚未创建要在虚拟化环境中运行的系统的映像，需要从安装媒体创建一个映像。在这种情况下，您需要准备一个硬盘映像，并获取安装媒体的映像或该媒体本身。

使用 `qemu-img` 创建硬盘。

```
tux > qemu-img create ❶ -f raw ❷ /images/sles/hda ❸ 8G ❹
```

- ❶ `create` 子命令告知 `qemu-img` 创建新映像。
- ❷ 使用 `-f` 参数指定磁盘的格式。
- ❸ 映像文件的完整路径。
- ❹ 映像大小，在本例中为 8 GB。该映像创建为稀疏映像文件，会随着数据填充到磁盘而增长。指定的大小定义映像文件可增长到的最大大小。

至少创建了一个硬盘映像后，您可以使用 `qemu-system-ARCH` 设置一个将引导到安装系统的虚拟机：

```
root # qemu-system-x86_64 -name "sles" ❶ -machine accel=kvm -M pc ❷ -m 768 ❸ \  
-smp 2 ❹ -boot d ❺ \  
-drive file=/images/sles/hda,if=virtio,index=0,media=disk,format=raw ❻ \  

```

```
-drive file=/isos/SLE-15-SP2-Online-ARCH-GM-medial.iso,index=1,media=cdrom ⑦ \  
-net nic,model=virtio,macaddr=52:54:00:05:11:11 ⑧ -net user \  
-vga cirrus ⑨ -balloon virtio ⑩
```

- ① 虚拟机的名称，将在窗口标题中显示，并用于 VNC 服务器。此名称必须是唯一的。
- ② 指定计算机类型。使用 `qemu-system-ARCH -M ?` 显示有效参数的列表。`pc` 是默认的标准 PC。
- ③ 虚拟机的最大内存量。
- ④ 定义包含两个处理器的 SMP 系统。
- ⑤ 指定引导顺序。有效值为 `a`、`b`（软盘 1 和 2）、`c`（第一个硬盘）、`d`（第一个 CD-ROM）或 `n` 到 `p`（从网络适配器 1-3 进行无盘引导）。默认值为 `c`。
- ⑥ 定义第一个 (`index=0`) 硬盘。系统会将该硬盘作为 `raw` 格式的半虚拟化 (`if=virtio`) 驱动器来访问。
- ⑦ 第二个 (`index=1`) 映像驱动器将充当 CD-ROM。
- ⑧ 定义 MAC 地址为 `52:54:00:05:11:11` 的半虚拟化 (`model=virtio`) 网络适配器。请务必指定唯一的 MAC 地址，否则会发生网络冲突。
- ⑨ 指定显卡。如果指定 `none`，将禁用显卡。
- ⑩ 定义允许动态更改内存量（最大为使用参数 `-m` 指定的最大值）的半虚拟化气球设备。

安装完 Guest 操作系统后，您无需指定 CD-ROM 设备即可启动相关的虚拟机：

```
root # 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
```

31.2 使用 `qemu-img` 管理磁盘映像

在上一节中（请参见第 31.1 节“使用 `qemu-system-ARCH` 进行基本安装”），我们使用 `qemu-img` 命令创建了硬盘的映像。另一方面，您可以使用 `qemu-img` 来执行一般的磁盘映像操作。本节介绍可帮助您灵活管理磁盘映像的 `qemu-img` 子命令。

31.2.1 有关 qemu-img 调用的一般信息

qemu-img（像 **zypper** 那样）使用子命令来执行特定的任务。每个子命令会识别一组不同的选项。有些选项是通用的，其中的多数子命令都可使用，而有些选项则专用于相关的子命令。有关所有受支持选项的列表，请参见 **qemu-img** 手册页 (**man 1 qemu-img**)。 **qemu-img** 使用以下一般语法：

```
tux > qemu-img subcommand [options]
```

支持以下子命令：

create

在文件系统中创建新磁盘映像。

check

检查现有磁盘映像是否有错误。

compare

检查两个映像的内容是否相同。

map

转储映像文件名的元数据及其后备文件链。

amend

修正映像文件名的映像格式特定选项。

convert

将现有磁盘映像转换为其他格式的新映像。

info

显示相关磁盘映像的信息。

snapshot

管理现有磁盘映像的快照。

commit

应用对现有磁盘映像进行的更改。

rebase

基于现有映像创建新的基本映像。

resize

增大或减小现有映像的大小。

31.2.2 创建、转换和检查磁盘映像

本节介绍如何创建磁盘映像、检查其状态、转换磁盘映像的格式，以及获取有关特定磁盘映像的详细信息。

31.2.2.1 qemu-img create

使用 **qemu-img create** 可为 VM Guest 操作系统创建新磁盘映像。该命令使用以下语法：

```
tux > qemu-img create -f fmt① -o options② fname③ size④
```

- ① 目标映像的格式。支持的格式为 raw 和 qcow2。
- ② 某些映像格式支持在命令行上传递其他选项。可在此处使用 -o 选项指定这些附加选项。raw 映像格式仅支持 size 选项，因此可以插入 -o size=8G，而不要在命令的末尾添加大小选项。
- ③ 要创建的目标磁盘映像的路径。
- ④ 目标磁盘映像的大小（如果尚未使用 -o size=<image_size> 选项指定）。映像大小的可选后缀为 K (KB)、M (MB)、G (GB) 或 T (TB)。

要在 /images 目录中创建最大可增长至 4 GB 的新磁盘映像 sles.raw，请运行以下命令：

```
tux > qemu-img create -f raw -o size=4G /images/sles.raw
Formatting '/images/sles.raw', fmt=raw size=4294967296

tux > ls -l /images/sles.raw
-rw-r--r-- 1 tux users 4294967296 Nov 15 15:56 /images/sles.raw

tux > qemu-img info /images/sles.raw
image: /images/sles11.raw
file format: raw
virtual size: 4.0G (4294967296 bytes)
disk size: 0
```

可以看到，新创建的映像的虚拟大小为 4 GB，但报告的实际磁盘大小为 0，因为尚未将任何数据写入该映像。



提示：Btrfs 文件系统上的 VM Guest 映像

如果您需要在 Btrfs 文件系统上创建磁盘映像，可以使用 `nocow=on` 来减少 Btrfs 的写入时复制功能产生的性能开销。

```
tux > qemu-img create -o nocow=on test.img 8G
```

但是，如果您想使用写入时复制（例如，要使用此功能来创建快照或者在虚拟机之间共享快照），请在命令行中省略 `nocow` 选项。

31.2.2.2 `qemu-img convert`

使用 `qemu-img convert` 可将磁盘映像转换为另一种格式。要获取 QEMU 支持的映像格式的完整列表，请运行 `qemu-img -h` 并查看输出的最后一行。该命令使用以下语法：

```
tux > qemu-img convert -c ❶ -f fmt ❷ -O out_fmt ❸ -o options ❹ fname ❺  
out_fname ❻
```

- ❶ 对目标磁盘映像应用压缩。只有 `qcow` 和 `qcow2` 格式支持压缩。
- ❷ 源磁盘映像的格式。系统通常会自动检测格式，因此可以省略此参数。
- ❸ 目标磁盘映像的格式。
- ❹ 指定目标映像格式相关的其他选项。使用 `-o ?` 可查看目标映像格式支持的选项列表。
- ❺ 要转换的源磁盘映像的路径。
- ❻ 转换后的目标磁盘映像的路径。

```
tux > qemu-img convert -O vmdk /images/sles.raw \  
/images/sles.vmdk  
  
tux > 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
```

要查看选定目标映像格式相关的选项列表，请运行以下命令（请将 `vmdk` 替换为您的映像格式）：

```
tux > 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
```

31.2.2.3 `qemu-img check`

使用 `qemu-img check` 可以检查现有磁盘映像是否有错误。并非所有磁盘映像格式都支持此功能。该命令使用以下语法：

```
tux > qemu-img check -f fmt❶ fname❷
```

- ❶ 源磁盘映像的格式。系统通常会自动检测格式，因此可以省略此参数。
- ❷ 要检查的源磁盘映像的路径。

如果未发现错误，该命令不返回任何输出，否则会显示所发现的错误的类型和数量。

```
tux > 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.
```

31.2.2.4 增大现有磁盘映像的大小

创建新映像时，必须在创建映像之前指定其最大大小（请参见第 31.2.2.1 节 “`qemu-img create`”）。安装 VM Guest 并使用一段时间后，映像的初始大小可能不再够用。在这种情况下，可向映像增加空间。

要将现有磁盘映像的大小增大 2 GB，请使用：

```
tux > qemu-img resize /images/sles.raw +2GB
```



注意

您可以调整 `raw` 和 `qcow2` 格式的磁盘映像的大小。要调整其他格式的映像的大小，请先使用 `qemu-img convert` 将其转换为支持的格式。

现在，该映像的最后一个分区后面包含 2 GB 可用空间。您可以调整现有分区的大小，或添加新分区。

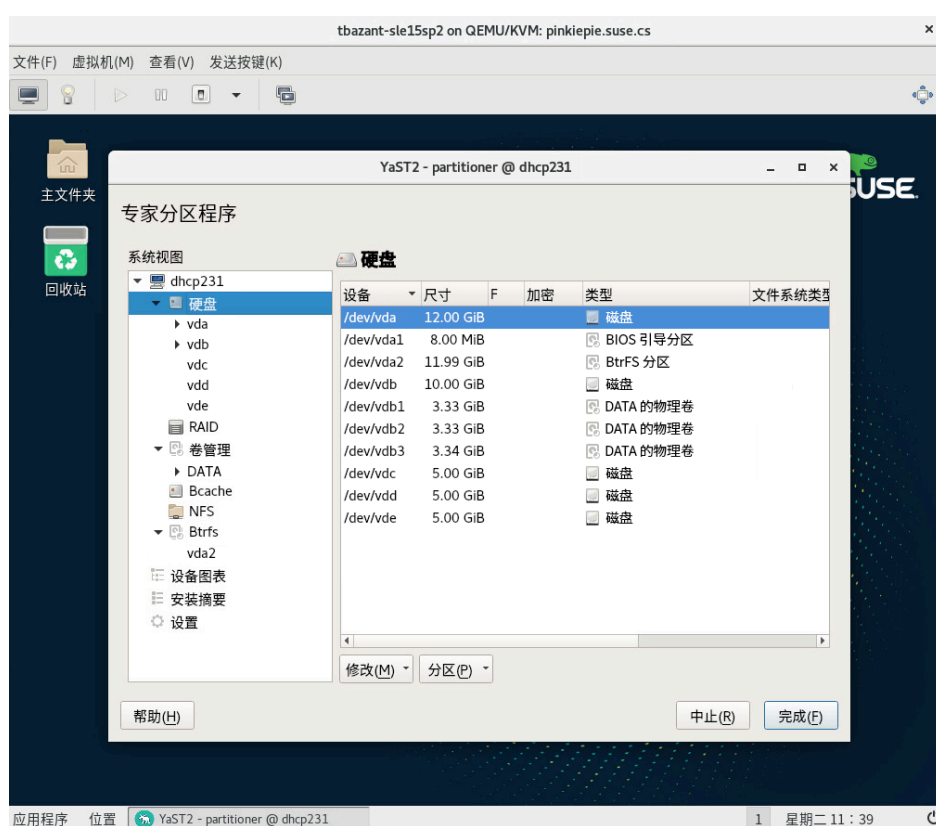


图 31.1：在 GUEST YAST 分区程序中新建 2 GB 分区

31.2.2.5 qcow2 文件格式的高级选项

qcow2 是 QEMU 使用的主要磁盘映像格式。其大小可按需增长，仅当虚拟机确实需要磁盘空间时才分配磁盘空间。

qcow2 格式的文件以恒定大小的单元进行组织。这些单元称为簇。从 Guest 的角度而言，虚拟磁盘也可划分为相同大小的簇。QEMU 默认为 64 kB 簇，但您可以在创建新映像时指定不同的值：

```
tux > qemu-img create -f qcow2 -o cluster_size=128K virt_disk.qcow2 4G
```

qcow2 映像包含一组表，这些表划分为两个级别，分别称为 L1 表和 L2 表。每个磁盘映像只有一个 L1 表，而根据映像的大小，L2 表可能有很多。

要在虚拟磁盘中读取或写入数据，QEMU 需要读取其对应的 L2 表，以确定相关数据位置。由于为每个 I/O 操作读取该表会消耗系统资源，QEMU 会在内存中缓存 L2 表，以提高磁盘访问速度。

31.2.2.5.1 选择适当的缓存大小

缓存大小与分配的空间量相关。L2 缓存可以映射以下虚拟磁盘空间量：

```
disk_size = l2_cache_size * cluster_size / 8
```

使用默认 64 kB 簇大小，即

```
disk_size = l2_cache_size * 8192
```

因此，要使缓存在使用默认簇大小的情况下映射 n GB 磁盘空间，需要

```
l2_cache_size = disk_size_GB * 131072
```

QEMU 默认使用 1 MB (1048576 字节) 的 L2 缓存。根据上面的公式，1 MB 的 L2 缓存涵盖了 8 GB (1048576 / 131072) 的虚拟磁盘空间。这意味着，如果您的虚拟磁盘大小不超过 8 GB，则使用默认 L2 缓存大小可使性能保持正常。如果磁盘更大，则可以通过增大 L2 缓存大小来提高磁盘访问速度。

31.2.2.5.2 配置缓存大小

可以在 QEMU 命令行上使用 -drive 选项来指定缓存大小。或者，可以在通过 QMP 通讯时使用 **blockdev-add** 命令。有关 QMP 的详细信息，请参见第 33.11 节“QMP - QEMU 计算机协议”。

以下选项配置虚拟 Guest 的缓存大小：

l2-cache-size

L2 表缓存的最大大小。

refcount-cache-size

refcount 块缓存的最大大小。有关 refcount 的详细信息，请参见 <https://raw.githubusercontent.com/qemu/qemu/master/docs/qcow2-cache.txt> 。

cache-size

上述两个缓存的合计最大大小。

指定上述选项的值时，请注意以下几点：

- L2 缓存和 refcount 块缓存的大小需是簇大小的倍数。
- 如果您仅设置其中一个选项，QEMU 将自动调整其他选项，使 L2 缓存比 refcount 缓存大 4 倍。

refcount 缓存的使用频率比 L2 缓存要低得多，因此您可以将 refcount 缓存设置得相对小一些：

```
root # qemu-system-ARCH [...] \  
-drive file=disk_image.qcow2,l2-cache-size=4194304,refcount-cache-size=262144
```

31.2.2.5.3 减小内存使用率

缓存越大，消耗的内存就越多。每个 qcow2 文件都有一个单独的 L2 缓存。使用大量较大的磁盘映像时，您可能需要相当大的内存量。如果您将后备文件（第 31.2.4 节“有效操作磁盘映像”）和快照（请参见第 31.2.3 节“使用 qemu-img 管理虚拟机的快照”）添加到 Guest 的设置链，则内存的消耗甚至更严重。

正因如此，QEMU 引入了 `cache-clean-interval` 设置。此设置定义一个以秒为单位的间隔，在此间隔过后，将从内存中去除未访问过的所有缓存项。

下面的示例每 10 分钟去除一次所有未使用的缓存项：

```
root # qemu-system-ARCH [...] -drive file=hd.qcow2,cache-clean-interval=600
```

如果未设置此选项，则默认值为 0，这会禁用此功能。

31.2.3 使用 qemu-img 管理虚拟机的快照

虚拟机快照是运行 VM Guest 的整个环境的快照。该快照包含处理器 (CPU)、内存 (RAM)、设备和所有可写磁盘的状态。

当您需要保存特定状态的虚拟机时，快照非常有用。例如，在虚拟化服务器上配置网络服务后，您可以从上次保存的虚拟机状态快速启动虚拟机。或者，您可以在关闭虚拟机之后创建快照，以便在尝试执行可能导致 VM Guest 不稳定的某种试验性操作之前创建备份状态。本节介绍后一种做法，前一种做法会在第 33 章 “使用 QEMU 监视器管理虚拟机” 中介绍。

要使用快照，您的 VM Guest 必须至少包含一个 `qcow2` 格式的可写硬盘映像。此设备通常是第一个虚拟硬盘。

虚拟机快照是在交互式 QEMU 监视器中使用 `savevm` 命令创建的。为了方便地识别特定的快照，可为其指派一个标记。有关 QEMU 监视器的详细信息，请参见第 33 章 “使用 QEMU 监视器管理虚拟机”。

`qcow2` 磁盘映像包含保存的快照后，您可以使用 `qemu-img snapshot` 命令检查这些快照。



警告：关闭 VM Guest

请不要在虚拟机正在运行时使用 `qemu-img snapshot` 命令创建或删除虚拟机快照。否则，可能会损坏包含保存的虚拟机状态的磁盘映像。

31.2.3.1 列出现有快照

使用 `qemu-img snapshot -l DISK_IMAGE` 可查看 `disk_image` 映像中保存的所有现有快照的列表。即使 VM Guest 正在运行，您也可以获取该列表。

```
tux > qemu-img snapshot -l /images/sles.qcow2
Snapshot list:
ID ①      TAG ②      VM SIZE ③      DATE ④      VM CLOCK ⑤
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
```

① 快照的唯一标识号。通常会自动递增。

- ② 快照的唯一说明字符串。它以直观易懂的 ID 形式来表示。
- ③ 快照占用的磁盘空间。请注意，运行中应用程序消耗的内存越多，快照就越大。
- ④ 快照的创建时间和日期。
- ⑤ 虚拟机时钟的当前状态。

31.2.3.2 创建已关闭虚拟机的快照

使用 **qemu-img snapshot -c** SNAPSHOT_TITLE DISK_IMAGE 可创建事先已关闭的虚拟机的当前状态快照。

```
tux > qemu-img snapshot -c backup_snapshot /images/sles.qcow2
```

```
tux > 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

如果某种情况干扰了 VM Guest 的运行，而您需要恢复到所保存快照（在本示例中为 ID 5）的状态，请关闭 VM Guest 并执行以下命令：

```
tux > qemu-img snapshot -a 5 /images/sles.qcow2
```

下一次您使用 **qemu-system-ARCH** 运行虚拟机时，它将处于编号为 5 的快照的状态。



注意

qemu-img snapshot -c 命令与 QEMU 监视器的 **savevm** 命令（请参见第 33 章“使用 QEMU 监视器管理虚拟机”）无关。例如，对于使用 **savevm** 在 QEMU 监视器中创建的快照，无法使用 **qemu-img snapshot -a** 应用快照。

31.2.3.3 删除快照

使用 **qemu-img snapshot -d SNAPSHOT_ID DISK_IMAGE** 可删除虚拟机的旧快照或不需要的快照。这可以节省 **qcow2** 磁盘映像中的一些磁盘空间，因为快照占用的空间将会恢复：

```
tux > qemu-img snapshot -d 2 /images/sles.qcow2
```

31.2.4 有效操作磁盘映像

假设存在以下真实情况：您是一名运行和管理多个虚拟化操作系统的服务器管理员。其中一组系统基于一个特定的发行套件，而另一组（或多个组）基于不同版本的发行套件，甚至不同的平台（也许不是 Unix）。使情况变得更复杂的是，基于同一发行套件的虚拟 Guest 系统通常会因部门和部署而各不相同。文件服务器使用的设置和服务通常与 Web 服务器不同，不过，两者可能都仍然基于 SUSE® Linux Enterprise Server。

使用 QEMU 可以创建“基本”磁盘映像。您可以将这些映像作为模板虚拟机使用。这些基本映像将为您节省大量时间，因为您将无需多次安装同一个操作系统。

31.2.4.1 基本映像和派生映像

首先，照常构建一个磁盘映像，并在其中安装目标系统。有关详细信息，请参见第 31.1 节“使用 **qemu-system-ARCH** 进行基本安装”和第 31.2.2 节“创建、转换和检查磁盘映像”。然后使用第一个映像作为基本映像来构建新映像。基本映像也称为后备文件。构建新的派生映像后，切勿再次引导基本映像，而是引导派生映像。多个派生映像可以同时依赖于一个基本映像。因此，更改基本映像可能会损坏依赖性。使用派生映像时，QEMU 会将更改写入其中，并仅使用基本映像进行读取操作。

比较好的做法是基于一个全新安装（并已根据需要注册）并且未应用任何补丁且未在其中安装或删除其他应用程序的操作系统创建基本映像。以后，您可以在应用最新补丁后基于原始基本映像创建另一个基本映像。

31.2.4.2 创建派生映像



注意

尽管您可以对基本映像使用 `raw` 格式，但不能对派生映像使用该格式，因为 `raw` 格式不支持 `backing_file` 选项。可对派生映像使用 `qcow2` 等格式。

例如，`/images/sles_base.raw` 是包含全新安装的系统的基本映像。

```
tux > 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
```

该映像的保留大小为 4 GB，实际大小为 2.4 GB，格式为 `raw`。使用以下命令创建自 `/images/sles_base.raw` 基本映像派生的映像：

```
tux > 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
```

查看派生映像的细节：

```
tux > 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)
```

尽管派生映像的保留大小与基本映像的大小相同 (4 GB)，但实际大小仅为 140 KB。原因是只有对派生映像内部的系统进行的更改会保存下来。运行派生的虚拟机，根据需要注册，并应用最新补丁。在系统中进行任何其他更改，例如，去除不需要的软件包或安装新软件包。然后关闭 VM Guest 并再次检查其细节：

```
tux > 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)
```

disk size 值已增长为 1.1 GB，这是文件系统（而不是基本映像）中的更改所占用的磁盘空间。

31.2.4.3 从派生映像重建基本映像

在修改派生映像（应用补丁、安装特定的应用程序、更改环境设置，等等）之后，它会达到所需的状态。此时，您可以合并原始基本映像和派生映像以创建新的基本映像。

原始基本映像 (/images/sles_base.raw) 包含全新安装的系统。它可以是经过修改的新基本映像的模板，而新的基本映像可以包含与第一个基本映像相同的系统，加上所有安全补丁和更新补丁等内容。在创建此新基本映像后，还可将它用作更专用的派生映像的模板。新基本映像便会独立于原始基本映像。基于派生映像创建基本映像的过程称为重建基本映像：

```
tux > qemu-img convert /images/sles_derived.qcow2 \
-O raw /images/sles_base2.raw
```

此命令创建了使用 raw 格式的新基本映像 /images/sles_base2.raw。

```
tux > 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
```

新映像比原始基本映像大 0.4 GB。它不使用任何后备文件，您可以轻松基于此映像创建新的派生映像。这样，您便可以为组织中的虚拟磁盘映像创建复杂的层次结构，并节省大量的时间和工作。

31.2.4.4 在 VM 主机服务器上装入映像

在主机系统下装入虚拟磁盘映像的做法可能会很实用。强烈建议阅读第 19 章 “libguestfs”，并使用专用的工具来访问虚拟机映像。不过，如果您需要手动执行此操作，请按照本指南所述操作。

Linux 系统可以使用回写设备装入 raw 磁盘映像的内部分区。第一个示例过程更复杂，但阐释得更清楚，而第二个过程则更简单直接：

过程 31.1：通过计算分区偏移量来装入磁盘映像

1. 在您要装入其分区的磁盘映像中设置一个循环设备。

```
tux > losetup /dev/loop0 /images/sles_base.raw
```

2. 确定您要装入的分区的扇区大小和起始扇区编号。

```
tux > fdisk -lu /dev/loop0
```

```
Disk /dev/loop0: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors
Units = sectors of 1 * 512 = 512 ❶ bytes
Disk identifier: 0x000ceca8
```

Device	Boot	Start	End	Blocks	Id	System
/dev/loop0p1		63	1542239	771088+	82	Linux swap
/dev/loop0p2	*	1542240 ❷	8385929	3421845	83	Linux

❶ 磁盘扇区大小。

❷ 分区的起始扇区。

3. 计算分区起始偏移量：

```
sector_size * sector_start = 512 * 1542240 = 789626880
```

4. 删除循环，并在准备好的目录中，根据计算出的偏移量装入磁盘映像中的分区。

```
tux > losetup -d /dev/loop0
tux > mount -o loop,offset=789626880 \
/images/sles_base.raw /mnt/sles/
tux > 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. 将一个或多个文件复制到装入的分区，并在完成后卸载该分区。

```
tux > cp /etc/X11/xorg.conf /mnt/sles/root/tmp
tux > ls -l /mnt/sles/root/tmp
tux > umount /mnt/sles/
```



警告：不要向当前正在使用的映像写入数据

切勿装入处于 读写 模式的运行中虚拟机映像的分区。这可能会损坏该分区并破坏整个 VM Guest。

32 使用 qemu-system-ARCH 运行虚拟机

准备好虚拟磁盘映像后（有关磁盘映像的详细信息，请参见第 31.2 节“使用 `qemu-img` 管理磁盘映像”），便可以启动相关的虚拟机了。第 31.1 节“使用 `qemu-system-ARCH` 进行基本安装”介绍了用于安装和运行 VM Guest 的简单命令。本章会重点详细解释 `qemu-system-ARCH` 的用法，并针对更具体的任务提供解决方案。有关 `qemu-system-ARCH` 的完整选项列表，请参见其手册页 (`man 1 qemu`)。

32.1 基本的 `qemu-system-ARCH` 调用

`qemu-system-ARCH` 命令使用以下语法：

```
qemu-system-ARCH options ❶ disk_img ❷
```

- ❶ `qemu-system-ARCH` 接受许多选项。其中的大部分选项定义模拟硬件的参数，其他选项会影响更一般性的模拟器行为。如果您不提供任何选项，则会使用默认值，在此情况下，您需要提供所要运行的磁盘映像的路径。
- ❷ 包含要虚拟化的 Guest 系统的磁盘映像路径。`qemu-system-ARCH` 支持许多映像格式。使用 `qemu-img --help` 可列出这些格式。如果您不以独立参数的形式提供磁盘映像的路径，则需要使用 `-drive file=` 选项。

32.2 常规 `qemu-system-ARCH` 选项

本节介绍常规 `qemu-system-ARCH` 选项，以及与基本模拟硬件（例如虚拟机的处理器、内存、型号类型或时间处理方法）相关的选项。

`-name NAME_OF_GUEST`

指定运行中 Guest 系统的名称。该名称将显示在窗口标题中，用于 VNC 服务器。

-boot OPTIONS

指定定义的驱动器的引导顺序。驱动器以字母（盘符）表示，a 和 b 代表软盘驱动器 1 和 2，c 代表第一个硬盘，d 代表第一个 CD-ROM 驱动器，n 到 p 代表无盘引导网络适配器。

例如，`qemu-system-ARCH [...] -boot order=ndc` 首先尝试从网络引导，然后尝试从第一个 CD-ROM 驱动器引导，最后尝试从第一个硬盘引导。

-pidfile FILENAME

将 QEMU 的进程标识号 (PID) 储存在文件中。如果您从脚本运行 QEMU，此文件非常有用。

-nodefaults

默认情况下，即使您不在命令行上指定基本虚拟设备，QEMU 也会创建这些设备。此选项会关闭此功能，在此情况下，您必须手动指定每个设备，包括显卡和网卡、并行或串行端口，或虚拟控制台。默认连 QEMU 监视器都不会挂接。

-daemonize

启动 QEMU 进程后将其“守护程序化”。在 QEMU 准备好接收其任何设备上的连接后，它会从标准输入和标准输出分离。



注意：SeaBIOS BIOS 实现

默认使用的 BIOS 是 SeaBIOS。您可以引导 USB 设备和任何驱动器（CD-ROM、软盘或硬盘）。SeaBIOS 支持 USB 鼠标和键盘，并支持多个 VGA 显卡。有关 SeaBIOS 的详细信息，请参见 [SeaBIOS 网站 \(https://www.seabios.org/SeaBIOS\)](https://www.seabios.org/SeaBIOS)。

32.2.1 基本虚拟硬件

32.2.1.1 计算机类型

您可以指定模拟计算机的类型。运行 `qemu-system-ARCH -M help` 可查看支持的计算机类型列表。



注意：ISA-PC

不支持 isapc: ISA-only-PC 计算机类型。

32.2.1.2 CPU 型号

要指定处理器 (CPU) 型号的类型，请运行 `qemu-system-ARCH -cpu MODEL`。使用 `qemu-system-ARCH -cpu help` 可查看支持的 CPU 型号列表。

[CPUID 维基百科 \(http://en.wikipedia.org/wiki/CPUID\)](http://en.wikipedia.org/wiki/CPUID) 中提供了 CPU 标志信息。

32.2.1.3 其他基本选项

下面是从命令行启动 qemu 时最常用的选项列表。要查看所有可用选项，请参见 `qemu-doc` 手册页。

`-m MEGABYTES`

指定用作虚拟 RAM 大小的 MB 数。

`-balloon virtio`

指定用于动态更改指派给 VM Guest 的虚拟 RAM 内存量的半虚拟化设备。上限是使用 `-m` 指定的内存量。

`-smp NUMBER_OF_CPUS`

指定要模拟的 CPU 数量。QEMU 在 PC 平台上最多支持 255 个 CPU（其中最多有 64 个 CPU 可使用 KVM 加速）。此选项还接受其他 CPU 相关的参数，例如插槽数、每个插槽的核心数，或每个核心的线程数。

下面是有效的 `qemu-system-ARCH` 命令行示例：

```
tux > qemu-system-x86_64 -name "SLES 12 SP2" -M pc-i440fx-2.7 -m 512 \
-machine accel=kvm -cpu kvm64 -smp 2 -drive format=raw,file=/images/sles.raw
```

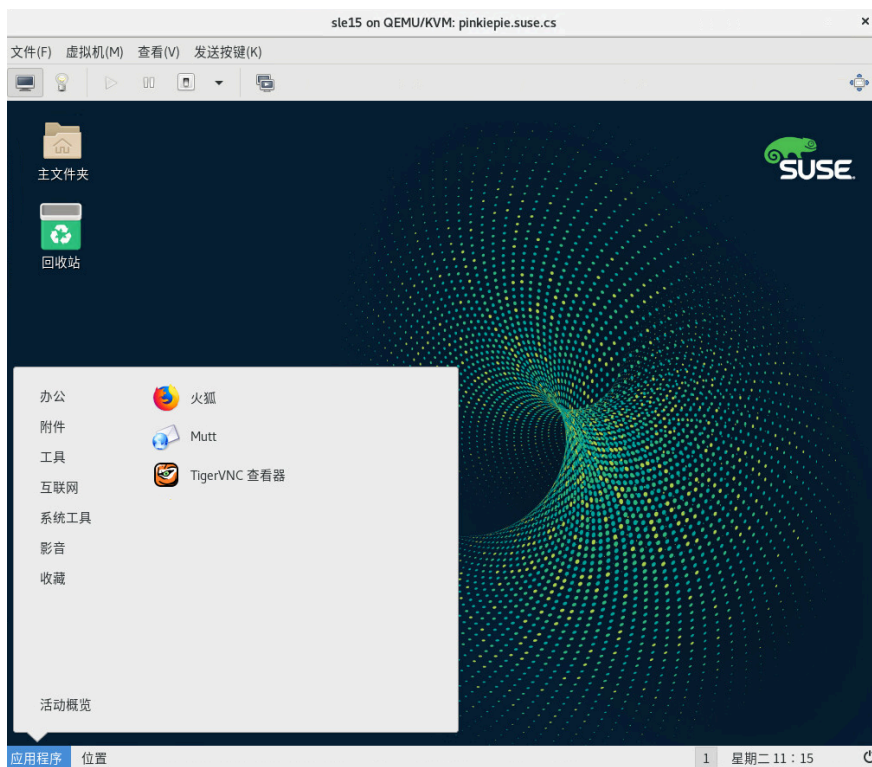



图 32.1：显示使用 SLES 作为 VM GUEST 的 QEMU 窗口

-no-acpi

禁用 ACPI 支持。

-S

QEMU 在 CPU 停止的状态下启动。要启动 CPU，请在 QEMU 监视器中输入 c。有关更多信息，请参见第 33 章“使用 QEMU 监视器管理虚拟机”。

32.2.2 储存和读取虚拟设备的配置

-readconfig CFG_FILE

您无需每次想要运行 VM Guest 时都在命令行上输入设备配置选项，**qemu-system-ARCH** 可以从先前使用 -writeconfig 保存的或者手动编辑的文件中读取相应配置。

-writeconfig CFG_FILE

将当前虚拟机的设备配置转储到文本文件，通过 -readconfig 选项可以重复使用该文件。

```
tux > qemu-system-x86_64 -name "SLES 12 SP2" -machine accel=kvm -M pc-
i440fx-2.7 -m 512 -cpu kvm64 \
-smp 2 /images/sles.raw -writeconfig /images/sles.cfg
(exited)
tux > cat /images/sles.cfg
# qemu config file

[drive]
  index = "0"
  media = "disk"
  file = "/images/sles_base.raw"
```

这样，您便可以有条不紊地有效管理虚拟机的设备配置。

32.2.3 Guest 实时时钟

-rtc OPTIONS

指定在 VM Guest 中处理 RTC 的方式。Guest 的时钟默认自主机系统的时钟派生。因此，建议将主机系统时钟与精确的外部时钟同步（例如，通过 NTP 服务同步）。

如果您需要将 VM Guest 时钟与主机时钟隔离，请指定 clock=vm，而不要使用默认值 clock=host。

您也可以使用 base 选项来指定 VM Guest 时钟的初始时间：

```
tux > qemu-system-x86_64 [...] -rtc clock=vm,base=2010-12-03T01:02:00
```

可以不指定时戳，而是指定 utc 或 localtime。前者指示 VM Guest 按当前 UTC（协调世界时，请参见 <http://en.wikipedia.org/wiki/UTC>）值启动，而后者则应用本地时间设置。

32.3 在 QEMU 中使用设备

QEMU 虚拟机会模拟运行 VM Guest 所需的所有设备。例如，QEMU 支持多种类型的网卡、块设备（硬盘和可移动驱动器）、USB 设备、字符设备（串行和并行端口）或多媒体设备（显卡和声卡）。本节介绍用于配置各种类型的受支持设备的选项。



提示

如果需要为设备（例如 `-drive`）设置特殊的驱动程序和驱动程序属性，请使用 `-device` 选项来指定，并使用 `drive=` 子选项进行标识。例如：

```
tux > sudo qemu-system-x86_64 [...] -drive if=none,id=drive0,format=raw \
-device virtio-blk-pci,drive=drive0,scsi=off ...
```

要获取有关可用驱动程序及其属性的帮助，请使用 `-device ?` 和 `-device DRIVER,?`。

32.3.1 块设备

块设备对于虚拟机而言至关重要。一般情况下，这些设备是固定或可移动的储存媒体，通常称作驱动器。通常会用连接的硬盘中的其中一块保存要虚拟化的 Guest 操作系统。

虚拟机驱动器使用 `-drive` 来定义。此选项具有许多子选项，本节将介绍其中的一些子选项。有关完整列表，请参见手册页 (`man 1 qemu`)。

`-drive` 选项的子选项

`file=image_fname`

指定要用于此驱动器的磁盘映像的路径。如果未指定，将使用一个空（可移动）驱动器。

`if=drive_interface`

指定驱动器要连接到的接口类型。SUSE 目前仅支持 `floppy`、`scsi`、`ide` 或 `virtio`。`virtio` 定义半虚拟化磁盘驱动程序。默认值为 `ide`。

`index=index_of_connector`

指定驱动器所连接到的磁盘接口（请参见 `if` 选项）上某个连接器的索引号。如果未指定，则索引会自动递增。

`media=type`

指定媒体的类型。可以是 `disk`（表示硬盘）或 `cdrom`（表示可移动的 CD-ROM 驱动器）。

format=img_fmt

指定连接的磁盘映像的格式。如果未指定，系统会自动检测格式。SUSE 目前支持 raw 和 qcow2 格式。

cache=method

指定驱动器的缓存方法。可能的值为

unsafe、writethrough、writeback、directsync 或 none。要在使用 qcow2 映像格式时提高性能，请选择 writeback。none 会禁用主机页缓存，因此是最安全的选项。对于映像文件，默认值为 writeback。有关更多信息，请参见第 17 章“磁盘缓存模式”。



提示

为了简化块设备的定义，QEMU 能够识别多种简写形式，以方便您输入 qemu-system-ARCH 命令行。

可使用

```
tux > sudo qemu-system-x86_64 -cdrom /images/cdrom.iso
```

来代替

```
tux > sudo qemu-system-x86_64 -drive format=raw,file=/images/cdrom.iso,index=2,media=cdrom
```

可使用

```
tux > sudo qemu-system-x86_64 -hda /images/image1.raw -hdb /images/image2.raw -hdc \
/images/image3.raw -hdd /images/image4.raw
```

来代替

```
tux > 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
```



提示：使用主机驱动器代替映像

作为使用磁盘映像（请参见第 31.2 节“使用 **qemu-img** 管理磁盘映像”）的替代方式，您还可以使用现有的 VM 主机服务器磁盘，将其作为驱动器进行连接，然后从 VM Guest 访问它们。请直接使用主机磁盘设备，而不要使用磁盘映像文件名。

要访问主机 CD-ROM 驱动器，请使用

```
tux > sudo qemu-system-x86_64 [...] -drive file=/dev/cdrom,media=cdrom
```

要访问主机硬盘，请使用

```
tux > sudo qemu-system-x86_64 [...] -drive file=/dev/hdb,media=disk
```

VM Guest 使用的主机驱动器不可同时由 VM 主机服务器或另一个 VM Guest 访问。

32.3.1.1 释放未使用的 Guest 磁盘空间

稀疏映像文件这种磁盘映像文件的大小会随着用户在其中添加数据而增长，它所占用的磁盘空间量等于其中储存的数据量。例如，如果您在稀疏磁盘映像中复制 1 GB 数据，则此映像的大小会增长 1 GB。如果您随后删除 500 MB（举例而言）的数据，映像大小默认不会按预期减小。

正因如此，KVM 命令行上引入了 `discard=on` 选项。此选项告知超级管理程序在从稀疏 Guest 映像中删除数据后自动释放“空洞”。请注意，此选项仅对 `if=scsi` 驱动器接口有效：

```
tux > sudo qemu-system-x86_64 [...] -drive format=img_format,file=/path/to/  
file.img,if=scsi,discard=on
```



重要：支持状态

不支持 `if=scsi`。此接口不会映射到 virtio-scsi，而是映射到 lsi SCSI 适配器。

32.3.1.2 IOThread

IOThread 是 virtio 设备的专用事件循环线程，用于执行 I/O 请求来提高可缩放性，尤其是在包含 SMP VM Guest 并使用许多磁盘设备的 SMP VM 主机服务器上。IOThread 不使用 QEMU 的主事件循环进行 I/O 处理，而是允许将 I/O 工作分散到多个 CPU 之间，经过正确配置后将可以改善延迟情况。

可通过定义 IOThread 对象来启用 IOThread。然后，virtio 设备可将这些对象用于其 I/O 事件循环。许多 virtio 设备都可以使用单个 IOThread 对象，或者可按 1:1 映射配置 virtio 设备和 IOThread 对象。下面的示例会创建 ID 为 `iothread0` 的单个 IOThread，然后，该 IOThread 将用作两个 virtio-blk 设备的事件循环。

```
tux > 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 [...]
```

下面的 qemu 命令行示例说明了 virtio 设备与 IOThread 之间的 1:1 映射：

```
tux > 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 [...]
```

32.3.1.3 virtio-blk 的基于 Bio 的 I/O 路径

为了提高 I/O 密集型应用程序的性能，内核版本 3.7 中为 virtio-blk 接口引入了新的 I/O 路径。这个基于 bio 的块设备驱动程序会跳过 I/O 调度程序，因此可缩短 Guest 中的 I/O 路径并降低延迟。对于 SSD 磁盘等高速储存设备，该驱动程序特别有用。

该驱动程序默认处于禁用状态。要使用该驱动程序，请执行以下操作：

1. 在 Guest 上的内核命令行中追加 `virtio_blk.use_bio=1`。可以通过 YaST > 系统 > 引导加载程序执行此操作。

为此，您还可以编辑 `/etc/default/grub`，搜索包含 `GRUB_CMDLINE_LINUX_DEFAULT=` 的行，并在末尾添加内核参数。然后运行 `grub2-mkconfig >/boot/grub2/grub.cfg` 以更新 grub2 引导菜单。

2. 在激活新内核命令行的情况下重引导 Guest。



提示：慢速设备上基于 Bio 的驱动程序

基于 bio 的 virtio-blk 驱动程序对于机械硬盘等慢速设备没有帮助。原因在于，调度所带来的优势大于缩短 bio 路径所带来的优势。请不要在慢速设备上使用基于 bio 的驱动程序。

32.3.1.4 直接访问 iSCSI 资源

QEMU 现已与 `libiscsi` 相集成。因此，QEMU 可以直接访问 iSCSI 资源并将其用作虚拟机块设备。此功能不需要任何主机 iSCSI 发起端配置，而基于 iSCSI 目标的 libvirt 储存池设置则需要这种配置。此功能通过用户空间库 `libiscsi` 直接将 Guest 储存接口连接到 iSCSI 目标 LUN。您也可以在 libvirt XML 配置中指定基于 iSCSI 的磁盘设备。



注意：RAW 映像格式

由于 iSCSI 协议存在一些技术方面的限制，仅当使用 RAW 映像格式时，此功能才可用。

下面是用于配置 iSCSI 连接的 QEMU 命令行界面。



注意：virt-manager 限制

virt-manager 界面尚未公开基于 `libiscsi` 的储存供应的用法，但是可以通过直接编辑 Guest XML 对其进行配置。这种访问基于 iSCSI 的储存的新方式通过命令行来实现。

```
tux > 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 ...
```

下面是使用基于协议的 iSCSI 的 Guest 域 XML 的示例代码段：


```
<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>
```

将此代码段与使用 virt-manager 设置的基于主机的 iSCSI 发起端示例相对比：

```
<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>
```

32.3.1.5 通过 QEMU 使用 RADOS 块设备

RADOS 块设备 (RBD) 将数据储存在 Ceph 群集中。这些设备支持快照、复制和数据一致性。您可以像使用其他块设备一样，从 KVM 管理的 VM Guest 使用 RBD。

有关更多细节，请参见《SUSE Enterprise Storage Administration Guide》（SUSE Enterprise Storage 管理指南）中的“Ceph as a Back-end for QEMU KVM Instance”（将 Ceph 用作 QEMU KVM 实例的后端）一章 (<https://documentation.suse.com/ses/6/html/ses-all/cha-ceph-kvm.html>) 。

32.3.2 图形设备和显示选项

本节介绍影响模拟视频卡类型的 QEMU 选项，以及 VM Guest 图形输出的显示方式。

32.3.2.1 定义视频卡

QEMU 使用 `-vga` 来定义用于显示 VM Guest 图形输出的视频卡。`-vga` 选项识别以下值：

none

在 VM Guest 上禁用视频卡（不模拟视频卡）。您仍可以通过串行控制台访问运行中的 VM Guest。

std

模拟标准的 VESA 2.0 VBE 视频卡。如果您打算在 VM Guest 上使用较高的显示分辨率，请使用此值。

cirrus

模拟 Cirrus Logic GD5446 视频卡。如果您需要确保模拟视频硬件的高兼容性，则此值非常适合。大多数操作系统（甚至包括 Windows 95）都能识别此类视频卡。



提示

为了在使用 `cirrus` 类型时获得最佳视频性能，请在 VM Guest 和 VM 主机服务器上都使用 16 位色深。

32.3.2.2 显示选项

以下选项会影响 VM Guest 图形输出的显示方式。

-display gtk

在 GTK 窗口中显示视频输出。此界面提供用于在运行时配置和控制 VM 的 UI 元素。

-display sdl

通过 SDL 显示视频输出（通常是在单独的图形窗口中）。有关详细信息，请参见 SDL 文档。

-spice option[,option[,...]]

启用 spice 远程桌面协议。

-display vnc

有关更多信息，请参考第 32.5 节“使用 VNC 查看 VM Guest”。

-nographic

禁用 QEMU 的图形输出。模拟的串行端口将重定向到控制台。

使用 -nographic 启动虚拟机后，在虚拟控制台中按 **Ctrl-A H** 可查看其他有用快捷键的列表，例如，用于在控制台与 QEMU 监视器之间切换的快捷键。

```
tux > 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

禁用 QEMU 窗口的装饰。便于在专用桌面工作空间中操作。

-full-screen

以全屏模式启动 QEMU 图形输出。

-no-quit

禁用 QEMU 窗口的关闭按钮，防止强行关闭窗口。

-alt-grab、-ctrl-grab

默认情况下，在按 **Ctrl** - **Alt** 之后，QEMU 窗口会释放“捕获的”鼠标。您可以将组合键更改为 **Ctrl** - **Alt** - **Shift** (-alt-grab) 或右 **Ctrl** 键 (-ctrl-grab)。

32.3.3 USB 设备

可通过两种方式来创建可供 KVM 中的 VM Guest 使用的 USB 设备：可以在 VM Guest 中模拟新的 USB 设备，或将现有的主机 USB 设备指派给 VM Guest。要在 QEMU 中使用 USB 设备，首先需要通过 -usb 选项启用通用 USB 驱动程序。然后可以通过 -usbdevice 选项指定各个设备。

32.3.3.1 在 VM Guest 中模拟 USB 设备

SUSE 目前支持以下类型的 USB 设

备：disk、host、serial、braille、net、mouse 和 tablet。

-usbdevice 选项的 USB 设备类型

disk

基于文件模拟大容量储存设备。可以使用可选的 format 选项，而不要检测格式。

```
tux > qemu-system-x86_64 [...] -usbdevice  
      disk:format=raw:/virt/usb_disk.raw
```

host

直通主机设备（由 bus.addr 标识）。

serial

主机字符设备的串行转换器。

braille

使用 BrlAPI 模拟盲文设备以显示盲文输出。

net

模拟支持 CDC 以太网和 RNDIS 协议的网络适配器。

mouse

模拟虚拟 USB 鼠标。此选项会覆盖默认的 PS/2 鼠标模拟。下面的示例显示了使用 `qemu-system-ARCH [...] -usbdevice mouse` 启动的 VM Guest 上的鼠标硬件状态：

```
tux > 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

模拟使用绝对坐标的定位设备（例如触摸屏）。此选项会覆盖默认的 PS/2 鼠标模拟。如果您要通过 VNC 协议查看 VM Guest，则绘图板设备非常有用。有关更多信息，请参见第 32.5 节“使用 VNC 查看 VM Guest”。

32.3.4 字符设备

使用 `-chardev` 可创建新的字符设备。该选项使用以下一般语法：

```
qemu-system-x86_64 [...] -chardev BACKEND_TYPE,id=ID_STRING
```

其中，`BACKEND_TYPE` 可以是

`null`、`socket`、`udp`、`msmouse`、`vc`、`file`、`pipe`、`console`、`serial`、`pty`、`stdio`、`braille`、`tty` 或 `parport`。所有字符设备都必须有一个最长为 127 个字符的唯一标识字符串。此字符串用于在其他相关指令中标识该设备。有关后端的所有子选项的完整说明，请参见手册页 (`man 1 qemu`)。下面是可用 后端 的简要说明：

null

创建一个空设备，该设备不输出数据且会丢弃收到的所有数据。

stdio

连接到 QEMU 的进程标准输入和标准输出。

socket

创建双向流套接字。如果指定了 PATH，则创建 Unix 套接字：

```
tux > sudo qemu-system-x86_64 [...] -chardev \  
socket,id=unix_socket1,path=/tmp/unix_socket1,server
```

SERVER 子选项指定该套接字是监听套接字。

如果指定了 PORT，则创建 TCP 套接字：

```
tux > sudo qemu-system-x86_64 [...] -chardev \  
socket,id=tcp_socket1,host=localhost,port=7777,server,nowait
```

该命令在端口 7777 上创建一个本地监听 (server) TCP 套接字。QEMU 将不会因为等待客户端连接到监听端口而进入阻塞状态 (nowait)。

udp

通过 UDP 协议将来自 VM Guest 的所有网络流量发送到远程主机。

```
tux > sudo qemu-system-x86_64 [...] \  
-chardev udp,id=udp_fwd,host=mercury.example.com,port=7777
```

该命令在远程主机 mercury.example.com 上绑定端口 7777，并从中发送 VM Guest 网络流量。

vc

创建新的 QEMU 文本控制台。您可以选择性地指定虚拟控制台的尺寸：

```
tux > sudo qemu-system-x86_64 [...] -chardev vc,id=vc1,width=640,height=480 \  
\   
-mon chardev=vc1
```

该命令会创建指定大小且名为 vc1 的新虚拟控制台，并将 QEMU 监视器连接到该控制台。

file

将来自 VM Guest 的所有流量都记录到 VM 主机服务器上的一个文件。必须指定 path，如果该路径不存在，系统将会予以创建。

```
tux > sudo qemu-system-x86_64 [...] \  
-chardev file,id=qemu_log1,path=/var/log/qemu/guest1.log
```

默认情况下，QEMU 将为串行与并行端口创建一组字符设备，并为 QEMU 监视器创建一个特殊控制台。不过，您可以创建自己的字符设备，并将其用于所述目的。以下选项可为您提供帮助：

-serial CHAR_DEV

将 VM Guest 的虚拟串行端口重定向到 VM 主机服务器上的字符设备 CHAR_DEV。在图形模式下，此设备默认是一个虚拟控制台 (vc)；在非图形模式下，默认则是 stdio。 -serial 可识别许多子选项。有关子选项的完整列表，请参见手册页 man 1 qemu。您最多可以模拟四个串行端口。使用 -serial none 可禁用所有串行端口。

-parallel DEVICE

将 VM Guest 的并行端口重定向到 DEVICE。此选项支持的设备与 -serial 相同。



提示

使用 SUSE Linux Enterprise Server 作为 VM 主机服务器时，您可以直接使用硬件并行端口设备 /dev/parportN（其中的 N 是端口号）。

您最多可以模拟三个并行端口。使用 -parallel none 可禁用所有并行端口。

-monitor CHAR_DEV

将 QEMU 监视器重定向到 VM 主机服务器上的字符设备 CHAR_DEV。此选项支持的设备与 -serial 相同。在图形模式下，此设备默认是一个虚拟控制台 (vc)；在非图形模式下，默认则是 stdio。

有关可用字符设备后端的完整列表，请参见手册页 (man 1 qemu)。

32.4 QEMU 中的网络

将 `-netdev` 选项与 `-device` 结合使用可为 VM Guest 定义特定类型的网络和网络接口卡。 `-netdev` 选项的语法为

```
-netdev type[,prop[=value][,...]]
```

SUSE 目前支持以下网络类型： `user`、 `bridge` 和 `tap`。有关 `-netdev` 子选项的完整列表，请参见手册页 (`man 1 qemu`)。

支持的 `-netdev` 子选项

`bridge`

使用指定的网络助手来配置 TAP 接口并将其挂接到指定的网桥。有关更多信息，请参见第 32.4.3 节 “桥接网络”。

`user`

指定用户模式网络。有关更多信息，请参见第 32.4.2 节 “用户模式网络”。

`tap`

指定桥接网络或路由网络。有关更多信息，请参见第 32.4.3 节 “桥接网络”。

32.4.1 定义网络接口卡

将 `-netdev` 与相关的 `-device` 选项搭配使用可以添加新的模拟网卡：

```
tux > sudo qemu-system-x86_64 [...] \  
-netdev tap①,id=hostnet0 \  
-device virtio-net-pci②,netdev=hostnet0,vlan=1③,\  
macaddr=00:16:35:AF:94:4B④,name=ncard1
```

- ① 指定网络设备类型。
- ② 指定网卡的型号。使用 `qemu-system-ARCH -device help` 并搜索 `Network devices`：部分可以获取您平台上受 QEMU 支持的所有网卡型号的列表。
SUSE 目前支持型号 `rtl8139`、`e1000` 及其衍生产品 `e1000-82540em`、`e1000-82544gc`、`e1000-82545em` 和 `virtio-net-pci`。要查看特定驱动程序的选项列表，请添加 `help` 作为驱动程序选项：

```
tux > sudo qemu-system-x86_64 -device e1000,help
e1000.mac=macaddr
e1000.vlan=vlan
e1000.netdev=netdev
e1000.bootindex=int32
e1000.autonegotiation=on/off
e1000.mitigation=on/off
e1000.addr=pci-devfn
e1000.romfile=str
e1000.rombar=uint32
e1000.multifunction=on/off
e1000.command_serr_enable=on/off
```

- ③ 将网络接口连接到 VLAN 1。您可以指定自己的编号，该编号主要用于标识目的。如果您省略此子选项，QEMU 将使用默认值 0。
- ④ 指定网卡的媒体访问控制 (MAC) 地址。它是一个唯一标识符，建议您始终指定该地址。如果未指定，QEMU 将提供自己的默认 MAC 地址，因此可能会在相关 VLAN 中造成 MAC 地址冲突。

32.4.2 用户模式网络

`-netdev user` 选项指示 QEMU 使用用户模式网络。如果未选择网络模式，则默认使用用户模式。因此，这些命令行等效于：

```
tux > sudo qemu-system-x86_64 -hda /images/sles_base.raw
```

```
tux > sudo qemu-system-x86_64 -hda /images/sles_base.raw -netdev
user,id=hostnet0
```

如果您要允许 VM Guest 访问外部网络资源（例如因特网），则此模式非常有用。默认不允许任何传入流量，因此 VM Guest 对于网络中的其他计算机不可见。在此网络模式下，将不需要管理员特权。用户模式还可用于从 VM 主机服务器上的本地目录在 VM Guest 上执行网络引导。

VM Guest 会获得虚拟 DHCP 服务器分配的一个 IP 地址。可通过 10.0.2.2 访问 VM 主机服务器（DHCP 服务器），而分配的 IP 地址范围从 10.0.2.15 开始。您可以使用 **ssh** 连接到 10.0.2.2 上的 VM 主机服务器，并使用 **scp** 来回复制文件。

32.4.2.1 命令行示例

本节提供了有关如何使用 QEMU 设置用户模式网络的几个示例。

例 32.1：受限用户模式网络

```
tux > sudo qemu-system-x86_64 [...] \  
-netdev user①,id=hostnet0 \  
-device virtio-net-pci,netdev=hostnet0,vlan=1②,name=user_net1③,restrict=yes④
```

- ① 指定用户模式网络。
- ② 连接到 VLAN 1。如果省略此选项，则默认使用 0。
- ③ 指定网络堆栈的直观易懂名称。可用于在 QEMU 监视器中标识该堆栈。
- ④ 隔离 VM Guest。这样 VM Guest 将无法与 VM 主机服务器通讯，并且网络包将不会路由到外部网络。

例 32.2：使用自定义 IP 范围的用户模式网络

```
tux > sudo qemu-system-x86_64 [...] \  
-netdev user,id=hostnet0 \  
-device virtio-net-pci,netdev=hostnet0,net=10.2.0.0/8①,host=10.2.0.6②,\  
dhcpstart=10.2.0.20③,hostname=tux_kvm_guest④
```

- ① 指定 VM Guest 看到的网络 IP 地址，以及可选的网络掩码。默认值为 10.0.2.0/8。
- ② 指定 VM Guest 看到的 VM 主机服务器 IP 地址。默认值为 10.0.2.2。
- ③ 指定可由内置 DHCP 服务器指派给 VM Guest 的 16 个 IP 地址中的第一个。默认值为 10.0.2.15。
- ④ 指定内置 DHCP 服务器将指派给 VM Guest 的主机名。

例 32.3：使用网络引导和 TFTP 的用户模式网络

```
tux > sudo qemu-system-x86_64 [...] \  
-netdev user,id=hostnet0 \  
-device virtio-net-pci,netdev=hostnet0,tftp=/images/tftp_dir①,\  
bootfile=/images/boot/pxelinux.0②
```

- ① 激活内置 TFTP（提供最为基本的 FTP 功能的文件传输协议）服务器。指定目录中的文件将以 TFTP 服务器根目录的形式显示给 VM Guest。

- ② 以 BOOTP（可提供引导映像 IP 地址和网络位置的一种网络协议，通常在无盘工作站中使用）文件的形式广播指定的文件。与 `tftp` 搭配使用时，可以通过主机上的本地目录从网络引导 VM Guest。

例 32.4：使用主机端口转发的用户模式网络

```
tux > sudo qemu-system-x86_64 [...] \  
-netdev user,id=hostnet0 \  
-device virtio-net-pci,netdev=hostnet0,hostfwd=tcp::2222-:22
```

将主机上端口 2222 的传入 TCP 连接转发到 VM Guest 上的端口 22 (SSH)。如果 `sshd` 正在 VM Guest 上运行，请输入

```
tux > ssh qemu_host -p 2222
```

（其中，`qemu_host` 是主机系统的主机名或 IP 地址），以获取 VM Guest 的 SSH 提示。

32.4.3 桥接网络

使用 `-netdev tap` 选项时，QEMU 会通过将主机 TAP 网络设备连接到 VM Guest 的指定 VLAN 来创建网桥。该网络设备的网络接口便会对网络的其余部分可见。此方法默认未启用，需要显式指定。

首先创建一个网桥，并将一个 VM 主机服务器物理网络接口（通常是 `eth0`）添加到其中：

1. 启动 YaST 控制中心并选择系统 > 网络设置。
2. 单击添加，然后从硬件对话框窗口的设备类型下拉框中选择网桥。单击下一步。
3. 选择您需要使用动态还是静态指派的 IP 地址，然后填写相关网络设置（如果适用）。
4. 在桥接的设备窗格中，选择要添加到网桥的以太网设备。
单击下一步。出现有关调整已配置设备的提示时，请单击继续。
5. 单击确定以应用更改。检查是否已创建网桥：

```
tux > bridge link  
2: eth0 state UP : <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master br0 \
```

```
state forwarding priority 32 cost 100
```

32.4.3.1 手动连接到网桥

使用以下示例脚本将 VM Guest 连接到新建的网桥接口 `br0`。脚本中的数个命令通过 `sudo` 机制运行，原因是这些命令需要 `root` 特权。



提示：要求的软件

要管理网桥，需要安装 `tunctl` 软件包。

```
#!/bin/bash
bridge=br0 ❶
tap=$(sudo tunctl -u $(whoami) -b) ❷
sudo ip link set $tap up ❸
sleep 1s ❹
sudo ip link add name $bridge type bridge
sudo ip link set $bridge up
sudo ip link set $tap master $bridge ❺
qemu-system-x86_64 -machine accel=kvm -m 512 -hda /images/sles_base.raw \
-netdev tap,id=hostnet0 \
-device virtio-net-pci,netdev=hostnet0,vlan=0,macaddr=00:16:35:AF:94:4B,\
ifname=$tap ❻,script=no ❼,downscript=no
sudo ip link set $tap nomaster ❽
sudo ip link set $tap down ❾
sudo tunctl -d $tap ❿
```

- ❶ 网桥设备的名称。
- ❷ 准备新的 TAP 设备并将其指派给运行脚本的用户。TAP 设备是常用于虚拟化和模拟设置的虚拟网络设备。
- ❸ 启动新建的 TAP 网络接口。
- ❹ 暂停 1 秒，以确保新 TAP 网络接口确实启动。
- ❺ 将新 TAP 设备添加到网桥 `br0`。
- ❻ `ifname=` 子选项指定用于桥接的 TAP 网络接口的名称。

- ⑦ **qemu-system-ARCH** 在连接到网桥之前，会检查 `script` 和 `downscript` 值。如果它在 VM 主机服务器文件系统上找到了指定的脚本，将会在连接到网桥之前运行 `script`，并在退出网络环境之后运行 `downscript`。您可以使用这些脚本来设置和拆除桥接接口。默认会检查 `/etc/qemu-ifup` 和 `/etc/qemu-ifdown`。如果指定了 `script=no` 和 `downscript=no`，则会禁用脚本执行功能，您需要手动处理其执行。
- ⑧ 删除网桥 `br0` 中的 TAP 接口。
- ⑨ 将 TAP 设备的状态设置为 `down`。
- ⑩ 拆除 TAP 设备。

32.4.3.2 使用 `qemu-bridge-helper` 连接到网桥

通过网桥将 VM Guest 连接到网络的另一种方式是使用 `qemu-bridge-helper` 助手程序。该程序可为您配置 TAP 接口并将其挂接到指定的网桥。默认的助手可执行文件为 `/usr/lib/qemu-bridge-helper`。该助手可执行文件的权限要求为 `setuid root`，也就是说，只允许虚拟化组 (`kvm`) 的成员执行。因此，**qemu-system-ARCH** 命令本身并不需要以 `root` 特权运行。

当您指定网桥时，会自动调用该助手：

```
qemu-system-x86_64 [...] \  
-netdev bridge,id=hostnet0,vlan=0,br=br0 \  
-device virtio-net-pci,netdev=hostnet0
```

您可以使用 `helper=/path/to/your/helper` 选项指定自己的自定义助手脚本来处理 TAP 设备配置（解除配置）：

```
qemu-system-x86_64 [...] \  
-netdev bridge,id=hostnet0,vlan=0,br=br0,helper=/path/to/bridge-helper \  
-device virtio-net-pci,netdev=hostnet0
```



提示

要定义对 `qemu-bridge-helper` 的访问特权，请检查 `/etc/qemu/bridge.conf` 文件。例如，以下指令

```
allow br0
```

允许 `qemu-system-ARCH` 命令将其 VM Guest 连接到网桥 `br0`。

32.5 使用 VNC 查看 VM Guest

默认情况下，QEMU 使用 GTK（一个跨平台工具包库）窗口来显示 VM Guest 的图形输出。如果指定了 `-vnc` 选项，您可以让 QEMU 监听指定的 VNC 显示器，并将其图形输出重定向到 VNC 会话。



提示

通过 VNC 会话操作 QEMU 的虚拟机时，使用 `-usbdevice tablet` 选项会很有用。

此外，如果您需要使用另一种键盘布局而不是默认的 `en-us`，请使用 `-k` 选项指定所需布局。

`-vnc` 的第一个子选项必须是 `display` 值。`-vnc` 选项识别以下 `display` 指定值：

`host:display`

只接受来自显示器编号 `display` 上的 `host` 的连接。随后运行 VNC 会话的 TCP 端口通常是值为 `5900 + display` 的数字。如果未指定 `host`，系统将接受来自任何主机的连接。

`unix:path`

VNC 服务器监听 Unix 域套接字上的连接。`path` 选项指定相关 Unix 套接字的位置。

`none`

将初始化 VNC 服务器功能，但不启动该服务器本身。您稍后可以使用 QEMU 监视器启动 VNC 服务器。有关更多信息，请参见第 33 章“使用 QEMU 监视器管理虚拟机”。

可以在 `display` 值的后面使用一个或多个选项标志（以逗号分隔）。有效选项为：

`reverse`

通过反向连接来连接监听方 VNC 客户端。

websocket

额外打开一个专用于 VNC Websocket 连接的 TCP 监听端口。根据定义，Websocket 端口为 5700+display。

password

要求对客户端连接使用基于口令的身份验证。

tls

要求客户端在与 VNC 服务器通讯时使用 TLS。

x509=/path/to/certificate/dir

指定了 TLS 时有效。要求使用 x509 身份凭证来协商 TLS 会话。

x509verify=/path/to/certificate/dir

指定了 TLS 时有效。要求使用 x509 身份凭证来协商 TLS 会话。

sasl

要求客户端使用 SASL 向 VNC 服务器进行身份验证。

acl

打开访问控制列表，以检查 x509 客户端证书和 SASL 参与方。

lossy

启用有损压缩方法（梯度、JPEG 等）。

non-adaptive

禁用自适应编码。默认会启用自适应编码。

share=[allow-exclusive|force-shared|ignore]

设置显示共享策略。



注意

有关显示选项的更多细节，请参见 [qemu-doc 手册页](#)。

VNC 示例用法：

```
tux > qemu-system-x86_64 [...] -vnc :5
# (on the client:)
```

```
wilber > vncviewer venus:5 &
```

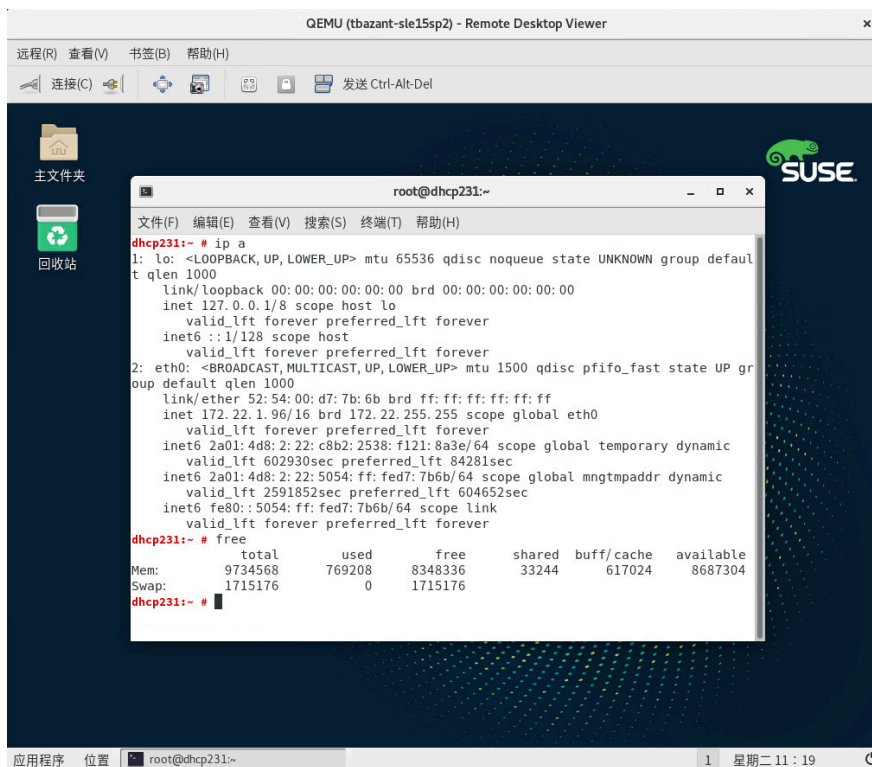


图 32.2：QEMU VNC 会话

32.5.1 保护 VNC 连接

默认的 VNC 服务器设置不使用任何形式的身份验证。在前面的示例中，任何用户都可以从网络中的任何主机连接和查看 QEMU VNC 会话。

系统提供了多个级别的安全性，可供您应用于 VNC 客户端/服务器连接。您可以使用口令、x509 证书、SASL 身份验证，甚至可在一条 QEMU 命令中结合数种身份验证方法来保护连接。

有关在 VM 主机服务器和客户端上配置 x509 证书的详细信息，请参见第 11.3.2 节“使用 x509 证书进行远程 TLS/SSL 连接 (qemu+tls 或 xen+tls)”和第 11.3.2.3 节“配置客户端并测试设置”。

Remmina VNC 查看器支持高级身份验证机制。因此，在以下示例中将使用该查看器来查看 VM Guest 的图形输出。对于此示例，我们假设服务器 x509 证书 `ca-cert.pem`、`server-cert.pem` 和 `server-key.pem` 位于主机上的 `/etc/pki/qemu` 目录中。可将客户端证书放在任何自定义目录中，Remmina 在连接启动时会要求提供这些证书的路径。

例 32.5：口令身份验证

```
qemu-system-x86_64 [...] -vnc :5,password -monitor stdio
```

在 VNC 显示器编号 5（通常为端口 5905）上启动 VM Guest 图形输出。`password` 子选项会初始化一种基于口令的简单身份验证方法。默认未设置口令，您需要在 QEMU 监视器中使用 `change vnc password` 命令设置一个口令：

```
QEMU 2.3.1 monitor - type 'help' for more information
(qemu) change vnc password
Password: ****
```

在此处需指定 `-monitor stdio` 选项，因为如果不重定向 QEMU 监视器的输入/输出，您将无法管理该监视器。

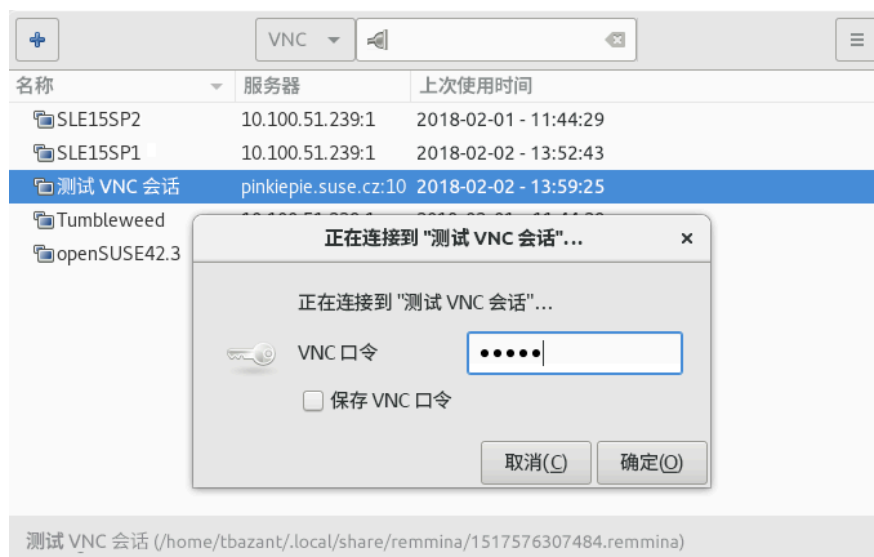


图 32.3：REMMINA 中的身份验证对话框

例 32.6：X509 证书身份验证

QEMU VNC 服务器可对会话使用 TLS 加密，并使用 x509 证书进行身份验证。服务器将要求客户端提供证书，并根据 CA 证书验证提供的证书。如果您的公司可提供内部证书颁发机构，请使用此身份验证类型。

```
qemu-system-x86_64 [...] -vnc :5,tls,x509verify=/etc/pki/qemu
```

例 32.7：X509 证书和口令身份验证

您可以将口令身份验证与 TLS 加密和 x509 证书身份验证结合使用，以便为客户端打造双层身份验证模型。运行以下命令后，请记得在 QEMU 监视器中设置口令：


```
qemu-system-x86_64 [...] -vnc :5,password,tls,x509verify=/etc/pki/qemu \
-monitor stdio
```

例 32.8：SASL 身份验证

简单身份验证和安全层 (SASL) 是因特网协议中的身份验证和数据安全性框架。它集成了多种身份验证机制，例如 PAM、Kerberos、LDAP 等等。SASL 会维护自己的用户数据库，因此 VM 主机服务器上无需存在连接用户帐户。

出于安全原因，建议您将 SASL 身份验证与 TLS 加密和 x509 证书结合使用：

```
qemu-system-x86_64 [...] -vnc :5,tls,x509,sasl -monitor stdio
```

33 使用 QEMU 监视器管理虚拟机

QEMU 运行时会提供一个监视器控制台用来与用户交互。使用监视器控制台中提供的命令可以检查运行中的操作系统、更改可移动媒体、抓取屏幕截图或音频片段，以及控制虚拟机的其他方面。



注意

下列章节列出了特选的实用 QEMU 监视器命令及其用途。要获取完整列表，请在 QEMU 监视器命令行中输入 `help`。

33.1 访问监视器控制台



提示：libvirt 没有监视器控制台

仅当您直接使用 `qemu-system-ARCH` 命令启动虚拟机并在本机 QEMU 窗口中查看其图形输出时，才可以访问监视器控制台。

如果您使用 `libvirt` 启动了虚拟机（例如，使用 `virt-manager`）并通过 VNC 或 Spice 会话查看其输出，则无法直接访问监视器控制台。不过，您可以通过 `virsh` 将监视器命令发送到虚拟机：

```
root # virsh qemu-monitor-command COMMAND
```

访问监视器控制台的方式取决于您使用哪种显示设备来查看虚拟机的输出。第 32.3.2.2 节“显示选项”中提供了有关显示器的更多细节。例如，要在使用 `-display gtk` 选项的情况下查看监视器，请按 `Ctrl-Alt-2`。同样，在使用 `-nographic` 选项时，可按组合键 `Ctrl-a c` 切换到监视器控制台。

在使用控制台时如需帮助，请使用 `help` 或 `?`。要获取有关特定命令的帮助，请使用 `help COMMAND`。

33.2 获取有关 Guest 系统的信息

要获取有关 Guest 系统的信息，请使用 **info**。如果不结合任何选项使用该命令，将列显可能的选项的列表。选项可确定要分析系统的哪个部分：

info version

显示 QEMU 的版本。

info commands

列出可用的 QMP 命令。

info network

显示网络状态。

info chardev

显示字符设备。

info block

有关块设备（例如硬盘、软盘驱动器或 CD-ROM）的信息。

info blockstats

块设备的读取和写入统计信息。

info registers

显示 CPU 寄存器。

info cpus

显示有关可用 CPU 的信息。

info history

显示命令行历史。

info irq

显示中断统计信息。

info pic

显示 i8259 (PIC) 状态。

info pci

显示 PCI 信息。

info tlb

显示虚拟内存到物理内存的映射。

info mem

显示活动的虚拟内存映射。

info jit

显示动态编译器信息。

info kvm

显示 KVM 信息。

info numa

显示 NUMA 信息。

info usb

显示 Guest USB 设备。

info usbhost

显示主机 USB 设备。

info profile

显示分析信息。

info capture

显示捕获（音频抓取）信息。

info snapshots

显示当前保存的虚拟机快照。

info status

显示当前虚拟机的状态。

info mice

显示哪些 Guest 鼠标正在接收事件。

info vnc

显示 VNC 服务器状态。

info name

显示当前虚拟机的名称。

info uuid

显示当前虚拟机的 UUID。

info usernet

显示用户网络堆栈连接状态。

info migrate

显示迁移状态。

info balloon

显示气球设备信息。

info qtree

显示设备树。

info qdm

显示 qdev 设备型号列表。

info roms

显示 ROM。

info migrate_cache_size

显示当前迁移 xbzrle（“基于 Xor 的零运行长度编码”）缓存大小。

info migrate_capabilities

显示各种迁移功能（例如 xbzrle 压缩）的状态。

info mtree

显示 VM Guest 内存层次结构。

info trace-events

显示可用的跟踪事件及其状态。

33.3 更改 VNC 口令

要更改 VNC 口令，请使用 `change vnc password` 命令并输入新口令：

```
(qemu) change vnc password
Password: *****
(qemu)
```

33.4 管理设备

要在 Guest 运行时添加新磁盘（热插入），请使用 `drive_add` 和 `device_add` 命令。首先定义要作为设备添加到总线 0 的新驱动器：

```
(qemu) drive_add 0 if=none,file=/tmp/test.img,format=raw,id=disk1
OK
```

可以通过查询块子系统来确认新设备：

```
(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
```

定义新驱动器后，需将它连接到某个设备，使 Guest 能够看到它。典型的设备是 `virtio-blk-pci` 或 `scsi-disk`。要获取可用的值的完整列表，请运行：

```
(qemu) device_add ?
name "VGA", bus PCI
name "usb-storage", bus usb-bus
[...]
name "virtio-blk-pci", bus virtio-bus
```

现在添加设备

```
(qemu) device_add virtio-blk-pci,drive=disk1,id=myvirtio1
```

并使用以下命令确认

```
(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 [0x0000ffe].
  id "myvirtio1"
```



提示

您可以使用 **device_del** 从 Guest 中去除通过 **device_add** 命令添加的设备。如需详细信息，请在 QEMU 监视器命令行上输入 **help device_del**。

要释放与可移动媒体设备连接的设备或文件，请使用 **eject DEVICE** 命令。使用可选的 **-f** 可以强制弹出。

要更改可移动媒体（例如 CD-ROM），请使用 **change DEVICE** 命令。可以使用 **info block** 命令确定可移动媒体的名称：

```
(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
```

33.5 控制键盘和鼠标

如果需要，可以使用监视器控制台来模拟键盘和鼠标输入。例如，如果您的图形用户界面会拦截某些低级别的组合键（例如 X Window 中的 **Ctrl - Alt - F1**），您仍可以使用 **sendkey KEYS** 来输入这些组合键：

```
sendkey ctrl-alt-f1
```

要列出 **KEYS** 选项中使用的按键名称，请输入 **sendkey** 并按 **→|**。

要控制鼠标，可使用以下命令：

mouse_move DX dy [DZ]

将活动的鼠标指针移到指定的坐标 dx, dy, dz（该滚动轴可选）。

mouse_button VAL

更改鼠标按钮的状态（1=左，2=中，4=右）。

mouse_set INDEX

设置由哪个鼠标设备接收事件。可以使用 **info mice** 命令获取设备索引号。

33.6 更改可用内存

如果启动虚拟机时使用了 **-balloon virtio** 选项（如此会启用半虚拟化气球设备），您便可以动态更改可用内存。有关启用气球设备的详细信息，请参见第 31.1 节“使用 **qemu-system-ARCH** 进行基本安装”。

要在监视器控制台中获取有关气球设备的信息并确定该设备是否已启用，请使用 **info balloon** 命令：

```
(qemu) info balloon
```

如果气球设备已启用，请使用 **balloon MEMORY_IN_MB** 命令设置请求的内存量：

```
(qemu) balloon 400
```

33.7 转储虚拟机内存

要将虚拟机内存的内容保存到磁盘或控制台输出，请使用以下命令：

memsave ADDR SIZE FILENAME

将起始地址为 **ADDR** 大小为 **SIZE** 的虚拟内存转储保存到 **FILENAME** 文件中

pmemsave ADDR SIZE FILENAME

将起始地址为 **ADDR** 大小为 **SIZE** 的物理内存转储保存到 **FILENAME** - 文件中

x / FMT ADDR

创建起始地址为 **ADDR** 并根据 **FMT** 字符串设置格式的虚拟内存转储。**FMT** 字符串由 **COUNTFORMATSIZE** 这三个参数构成：

COUNT 参数是要转储的项数。

FORMAT 可以是 x（十六进制）、d（有符号十进制）、u（无符号十进制）、o（八进制）、c（字符）或 i（汇编指令）。

SIZE 参数可以是 b（8 位）、h（16 位）、w（32 位）或 g（64 位）。在 x86 上，可以使用 i 格式指定 h 或 w，以分别选择 16 位或 32 位代码指令大小。

xp / FMT ADDR

创建起始地址为 ADDR 并根据 FMT 字符串设置格式的物理内存转储。FMT 字符串由 COUNTFORMATSIZE 这三个参数构成：

COUNT 参数是要转储的项数。

FORMAT 可以是 x（十六进制）、d（有符号十进制）、u（无符号十进制）、o（八进制）、c（字符）或 i（汇编指令）。

SIZE 参数可以是 b（8 位）、h（16 位）、w（32 位）或 g（64 位）。在 x86 上，可以使用 i 格式指定 h 或 w，以分别选择 16 位或 32 位代码指令大小。

33.8 管理虚拟机快照

SUSE 目前未正式支持在 QEMU 监视器中管理快照。本节中的信息在特定的情形下可能有帮助。

虚拟机快照是整个虚拟机的快照，包括 CPU、RAM 的状态以及所有可写磁盘的内容。要使用虚拟机快照，您必须至少有一个使用 qcow2 磁盘映像格式且不可移动的可写块设备。

当您需保存特定状态的虚拟机时，快照非常有用。例如，在虚拟化服务器上配置网络服务后，您可以从上次保存的虚拟机状态快速启动虚拟机。您还可以在关闭虚拟机之后创建快照，以便在尝试执行可能导致 VM Guest 不稳定的某种试验性操作之前创建备份状态。本节介绍前一种做法，后一种做法已在第 31.2.3 节“使用 qemu-img 管理虚拟机的快照”中介绍。

可在 QEMU 监视器中使用以下命令管理快照：

savevm NAME

创建一个标记为 NAME 的新虚拟机快照，或替换现有快照。

loadvm NAME

装载标记为 NAME 的虚拟机快照。

delvm

删除虚拟机快照。

info snapshots

列显有关可用快照的信息。

```
(qemu) info snapshots
Snapshot list:
ID ①      TAG ②      VM SIZE ③    DATE ④      VM CLOCK ⑤
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
```

- ① 快照的唯一标识号。通常会自动递增。
- ② 快照的唯一说明字符串。它以直观易懂的 ID 形式来表示。
- ③ 快照占用的磁盘空间。请注意，运行中应用程序消耗的内存越多，快照就越大。
- ④ 快照的创建时间和日期。
- ⑤ 虚拟机时钟的当前状态。

33.9 挂起和恢复虚拟机执行

以下命令可用于挂起和恢复虚拟机：

stop

挂起虚拟机的执行。

cont

恢复虚拟机的执行。

system_reset

重置虚拟机。效果类似于物理机上的复位按钮。这可能会使文件系统处于一种不干净状态。

system_powerdown

向计算机发送 **ACPI** 关机请求。效果类似于物理机上的电源按钮。

q 或 quit

立即终止 QEMU。

33.10 实时迁移

实时迁移过程可将任何虚拟机从一个主机系统传输到另一个主机系统，而不会对可用性造成任何干扰。您可以永久性更改主机，或者仅在维护期间更改主机。

实时迁移的要求：

- 第 10.7.1 节 “迁移要求” 中所述的所有要求均适用。
- 只能在具有相同 CPU 功能的 VM 主机服务器之间进行实时迁移。
- AHCI 接口、VirtFS 功能和 `-mem-path` 命令行选项与迁移不兼容。
- 必须以相同的方式启动源主机和目标主机上的 Guest。
- 不应使用 `-snapshot qemu` 命令行选项进行迁移（不支持此 `qemu` 命令行选项）。

！ 重要：支持状态

SUSE Linux Enterprise Server 中尚不支持 `postcopy` 模式。此模式仅发布为技术预览版。有关 `postcopy` 的详细信息，请参见 <http://wiki.qemu.org/Features/PostCopyLiveMigration>。

以下网站上提供了更多建议：<http://www.linux-kvm.org/page/Migration>

实时迁移过程包括以下步骤：

1. 虚拟机实例正在源主机上运行。
2. 虚拟机以冻结监听模式在目标主机上启动。使用的参数与源主机上相同，不过还要加上 `-incoming tcp:IP:PORT` 参数，其中的 `IP` 指定 IP 地址，`PORT` 指定用于监听传入迁移的端口。如果设置了 0 作为 IP 地址，则虚拟机将监听所有接口。
3. 在源主机上，切换到监视器控制台，并使用 `migrate -d tcp:DESTINATION_IP:PORT` 命令启动迁移。

4. 要确定迁移状态，请在源主机上的监视器控制台中使用 `info migrate` 命令。
5. 要取消迁移，请在源主机上的监视器控制台中使用 `migrate_cancel` 命令。
6. 要设置迁移时容许的最长停机时间（以秒为单位），请使用 `migrate_set_downtime` `NUMBER_OF_SECONDS` 命令。
7. 要设置最大迁移速度（每秒字节数），请使用 `migrate_set_speed` `BYTES_PER_SECOND` 命令。

33.11 QMP - QEMU 计算机协议

QMP 是基于 JSON 的协议，可使应用程序（例如 `libvirt`）能够与运行中的 QEMU 实例通讯。您可以使用 QMP 命令以多种方式访问 QEMU 监视器。

33.11.1 通过标准输入/输出访问 QMP

最灵活的使用 QMP 的方式是指定 `-mon` 选项。下面的示例使用标准输入/输出创建了一个 QMP 实例。请注意，在以下示例中，`->` 标记了包含从客户端发送到运行中 QEMU 实例的命令的行，而 `<-` 标记了包含 QEMU 返回的输出的行。

```
tux > 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": [  

```

```
    ]
  }
}
```

建立新的 QMP 连接后，QMP 将发送其问候消息并进入功能协商模式。在此模式下，只有 **qmp_capabilities** 命令能够正常运行。要退出功能协商模式并进入命令模式，必须先发出 **qmp_capabilities** 命令：

```
-> { "execute": "qmp_capabilities" }
<- {
  "return": {
  }
}
```

请注意，`"return": {}` 是 QMP 的成功响应。

QMP 的命令可以附带参数。例如，要弹出 CD-ROM 驱动器，请输入以下命令：

```
->{ "execute": "eject", "arguments": { "device": "ide1-cd0" } }
<- {
  "timestamp": {
    "seconds": 1410353381,
    "microseconds": 763480
  },
  "event": "DEVICE_TRAY_MOVED",
  "data": {
    "device": "ide1-cd0",
    "tray-open": true
  }
}
{
  "return": {
  }
}
```

33.11.2 通过 Telnet 访问 QMP

如果不使用标准输入/输出，您可将 QMP 接口连接到某个网络套接字，并通过特定的端口与其通讯：

```
tux > sudo qemu-system-x86_64 [...] \  
-chardev socket,id=mon0,host=localhost,port=4444,server,nowait \  
-mon chardev=mon0,mode=control,pretty=on
```

然后运行 telnet 连接到端口 4444:

```
tux > telnet localhost 4444  
Trying ::1...  
Connected to localhost.  
Escape character is '^]'.  
<- {  
  "QMP": {  
    "version": {  
      "qemu": {  
        "micro": 0,  
        "minor": 0,  
        "major": 2  
      },  
      "package": ""  
    },  
    "capabilities": [  
  ]  
}  
}
```

您可以同时创建多个监视器接口。下面的示例在标准输入/输出上创建了一个 HMP 实例（可识别“常规”QEMU 监视器命令的人工监视器），并在本地主机端口 4444 上创建了一个 QMP 实例：

```
tux > 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
```

33.11.3 通过 Unix 套接字访问 QMP

使用 `-qmp` 选项调用 QEMU，并创建一个 Unix 套接字：

```
tux > sudo qemu-system-x86_64 [...] \  
-qmp unix:/tmp/qmp-sock,server --monitor stdio  
  
QEMU waiting for connection on: unix:./qmp-sock,server
```

要通过 `/tmp/qmp-sock` 套接字来与 QEMU 实例通讯，请在同一主机上的另一个终端中使用 `nc`（有关详细信息，请参见 [man 1 nc](#)）：

```
tux > sudo nc -U /tmp/qmp-sock  
<- {"QMP": {"version": {"qemu": {"micro": 0, "minor": 0, "major": 2} [...]}}
```

33.11.4 通过 libvirt 的 `virsh` 命令访问 QMP

如果您在 `libvirt` 下运行虚拟机（请参见第 II 部分“使用 `libvirt` 管理虚拟机”），则可以通过运行 `virsh qemu-monitor-command` 来与它的运行中 Guest 通讯：

```
tux > sudo virsh qemu-monitor-command vm_guest1 \  
--pretty '{"execute":"query-kvm"}'  
<- {  
  "return": {  
    "enabled": true,  
    "present": true  
  },  
  "id": "libvirt-8"  
}
```

在上面的示例中，我们运行了简单的 `query-kvm` 命令来检查主机是否能够运行 KVM，以及是否启用了 KVM。



提示：生成直观易懂的输出

要使用 QEMU 的直观易懂的标准输出格式而不是 JSON 格式，请使用 `--hmp` 选项：

```
tux > sudo virsh qemu-monitor-command vm_guest1 --hmp "query-kvm"
```

VI 使用 LXC 管理虚拟机

34 Linux 容器 329

35 从 LXC 迁移到 libvirt-lxc 336

34 Linux 容器

34.1 设置 LXC 发行套件容器

容器是包含应用程序的代码及其所有依赖项的自包含软件。容器化应用程序可以快速完成部署，并可在计算环境中可靠运行。

要设置 LXC 容器，需要创建一个包含 Guest 发行套件的根文件系统。

过程 34.1：创建根文件系统

目前没有任何 GUI 可用于创建根文件系统。请以 root 身份运行 **`virt-create-rootfs`** 命令，以设置新的根文件系统。执行以下步骤在 `/path/to/rootfs` 中创建新的根文件系统。



重要：需要注册代码

`virt-create-rootfs` 需要使用注册代码来设置 SUSE Linux Enterprise Server 根文件系统。

1. 运行 **`virt-create-rootfs`** 命令：

```
tux > virt-create-rootfs --root /PATH/TO/ROOTFS --distro SLES-12.0 -  
c REGISTRATION_CODE
```

2. 使用 **`chroot`** 命令更改根文件系统的根路径：

```
tux > chroot /path/to/rootfs
```

3. 使用 **`passwd`** 更改 `root` 用户的口令。

4. 创建一个不具有 `root` 特权的 `操作员` 用户：

```
useradd -m operator
```

5. 更改操作员的口令：

```
passwd operator
```

6. 使用 `exit` 退出 chroot 环境。

过程 34.2：定义容器

1. 启动虚拟机管理器。
2. （可选）单击文件 > 添加连接以添加一个本地 LXC 连接（如果尚不存在）。
选择 LXC（Linux 容器）作为超级管理程序，然后单击连接。
3. 选择 localhost (LXC) 连接并单击文件 新建虚拟机菜单。
4. 激活操作系统容器并单击前进。
5. 键入过程 34.1 “创建根文件系统”中所述的根文件系统路径，然后单击前进按钮。
6. 选择要分配给该容器的最大内存量和最大 CPU 数量。然后单击前进按钮。
7. 键入容器的名称。对容器运行的所有 `virsh` 命令都将使用此名称。
单击高级选项。选择要将容器连接到的网络，然后单击完成按钮：系统随即会创建并启动该容器。此外一个控制台会自动打开。

过程 34.3：配置网络接口的 IP 地址

对于网络设备以及具有网络功能的 hostdev 设备，系统可能会为其提供要在 Guest 中的网络设备上设置的一个或多个 IP 地址。但某些超级管理程序或网络设备会直接忽略这些 IP 地址，或者只使用第一个 IP 地址。

1. 使用 `virsh` 编辑容器 XML 配置：

```
tux > virsh -c lxc:/// edit MYCONTAINER
```

2. 下面的示例说明如何设置一个或多个 IP 地址：

```
[...]  
<devices>  
  <interface type='network'>  
    <source network='default' />  
    <target dev='vnet0' />
```

```

<ip address='192.168.122.5' prefix='24' />
<ip address='192.168.122.5' prefix='24' peer①='10.0.0.10' />
  <route family②='ipv4' address③='192.168.122.0' prefix④='24'
    gateway⑤='192.168.122.1' />
  <route family②='ipv4' address③='192.168.122.8'
gateway⑤='192.168.122.1' />
</interface>
[...]
<hostdev mode='capabilities' type='net'>
  <source>
    <interface>eth0</interface>
  </source>
  <ip address='192.168.122.6' prefix='24' />
  <route family='ipv4' address='192.168.122.0' prefix='24'
gateway='192.168.122.1' />
  <route family='ipv4' address='192.168.122.8' gateway='192.168.122.1' />
</hostdev>
</devices>
[...]

```

- ① 可选属性。包含点对点网络设备另一端的 IP 地址。
- ② 可设置为 `ipv4` 或 `ipv6`。
- ③ 包含 IP 地址。
- ④ 可选参数（如果不指定，则会自动设置）。定义网络掩码中的 1 位的数量。对于 IPv4，将根据网络“类别”（`A`、`B` 或 `C`）确定默认前缀。对于 IPv6，默认前缀为 `64`。
- ⑤ 如果您未在 XML 文件中指定默认网关，将不会设置网关。

3. 您还可以添加 `route` 元素来定义要在 Guest 中添加的 IP 路由。这些路由将由 LXC 驱动程序使用。

```

[...]
<devices>
  <interface type①='ethernet'>
    <source>②
      <ip address③='192.168.123.1' prefix='24' />
      <ip address④='10.0.0.10' prefix='24' peer='192.168.122.5' />
    </source>
  </interface>
</devices>

```

```

<route⑤ family='ipv4' address='192.168.42.0' prefix='24'
      gateway='192.168.123.4' />
</source>
[...]
</interface>
[...]
</devices>
[...]

```

- ① 对于 以太网 类型的网络设备，可以选择性地为其提供要在网络设备主机端设置的一个或多个 IP 地址（③、④），以及一个或多个路由（⑤）。
- 这些 IP 地址和路由将配置为接口的 source 元素（②）的子元素。它们的属性与用于配置接口 Guest 端的名称类似的元素的属性相同（请参见上一步）。
- ③ 以太网 类型的网络设备的第一个 IP 地址。
- ④ 以太网 类型的网络设备的第二个 IP 地址。
- ⑤ 要在网络设备主机端设置的路由。

<http://libvirt.org/formatnetwork.html#elementsStaticroute> 上提供了有关此元素的属性的更多细节。

4. 保存更改并退出编辑器。



注意：容器网络

要配置容器网络，请编辑 `/etc/sysconfig/network/ifcfg-*` 文件。

34.2 设置 LXC 应用程序容器

Libvirt 还允许在容器中仅运行应用程序，而无需运行庞大的 Linux 发行套件。在此示例中，**bash** 将在自己的容器中启动。

过程 34.4：使用 YAST 定义应用程序容器

1. 启动虚拟机管理器。
2. （可选）单击文件 > 添加连接以添加一个本地 LXC 连接（如果尚不存在）。

选择 LXC（Linux 容器）作为超级管理程序，然后单击连接。

3. 选择 localhost (LXC) 连接并单击文件 新建虚拟机菜单。

4. 激活应用程序容器并单击前进。

设置要启动的应用程序的路径。例如，在字段中填充 /bin/sh，这样就可以创建第一个容器。单击前进。

5. 选择要分配给该容器的最大内存量和最大 CPU 数量。单击前进。

6. 键入容器的名称。对容器运行的所有 virsh 命令都将使用此名称。

单击高级选项。选择要将容器连接到的网络，然后单击完成。系统即会创建并启动该容器。一个控制台将会自动打开。

请注意，当应用程序运行完后，系统会销毁该容器。

34.3 使用 AppArmor 保护容器

默认未使用 AppArmor 或 SELinux 来保护容器。没有任何图形用户界面可用来更改 libvirt 域的安全模型，但 virsh 可提供帮助。

1. 使用 virsh 编辑容器 XML 配置：

```
tux > virsh -c lxc:/// edit MYCONTAINER
```

2. 将以下内容添加到 XML 配置中，然后保存配置并退出编辑器。

```
<domain>
...
<seclabel type="dynamic" model="apparmor"/>
...
</domain>
```

3. 设置此配置，系统将会在 /etc/apparmor.d/libvirt 目录中为容器创建 AppArmor 配置文件。默认配置文件仅允许最少量的应用程序在容器中运行。可以通过修改 libvirt-CONTAINER-uuid 文件来更改此设置：libvirt 不会重写此文件。

34.4 libvirt LXC 驱动程序与 LXC 之间的差异

SUSE Linux Enterprise Server 11 SP3 随附的是 LXC，而 SUSE Linux Enterprise Server 12 随附的是 libvirt LXC 驱动程序（为避免混淆，有时称为 libvirt-lxc）。容器在这些工具中的管理或配置方式是不同的。下面列出了部分差异。

主要差异在于，libvirt 中的域配置是一个 XML 文件，而 LXC 配置是一个属性文件。大多数 LXC 属性都可映射到域 XML。无法迁移的属性如下：

- `lxc.network.script.up`：可以使用 `/etc/libvirt/hooks/network` libvirt 钩子实现此脚本，不过需要调整该脚本。
- `lxc.network.ipv*`：libvirt 无法基于域配置设置容器网络配置。
- `lxc.network.name`：libvirt 无法设置容器网卡名称。
- `lxc.devtttydir`：libvirt 不允许更改控制台设备的位置。
- `lxc.console`：目前无法将控制台输出记录到 libvirt LXC 容器主机上的文件中。
- `lxc.pivotdir`：libvirt 不允许微调用 `lxc.pivotdir` 的目录。将使用 `/olroot`。
- `lxc.rootfs.mount`：libvirt 不允许微调此项。

LXC VLAN 网络会自动在主机上创建 VLAN 接口，然后将其移入 Guest 名称空间。libvirt-lxc 配置只能为 Open vSwitch tap 设备或 SR-IOV VF 的 PCI 直通指出 VLAN 标记 ID。转换工具实际上需要用户手动在主机端创建 VLAN 接口。

LXC rootfs 也可以是映像文件，但 LXC 会强行突破装入点，以尝试检测适当的文件系统格式。libvirt-lxc 可以装入多种格式的映像文件，但不显式支持格式参数的“auto”值。这意味着，在此情况下，用户需要优化生成的配置才能获得适当的匹配项。

LXC 支持任何 cgroup 配置（即使是将来的配置），而 libvirt 域配置需要映射每个 cgroup 配置。

LXC 可以在 rootfs 中装入块设备，但无法装入原始分区文件：需要手动将文件挂接到循环设备。相比之下，libvirt-lxc 既可以装入块设备，也可以装入任何格式的分区文件。

34.5 跨容器共享名称空间

与 Docker 开源引擎一样，libvirt 可让您从容器或进程继承名称空间，以共享网络名称空间。下面的示例说明如何共享所需的名称空间。

```
<domain type='lxc' xmlns:lxc='http://libvirt.org/schemas/domain/lxc/1.0'>
  [...]
  <lxc:namespace>
    <lxc:sharenet type='netns' value='red' />
    <lxc:shareuts type='name' value='CONTAINER_1' />
    <lxc:shareipc type='pid' value='12345' />
  </lxc:namespace>
</domain>
```

netns 选项特定于 sharenet。可通过此选项使用现有的网络名称空间（无需为容器创建新的网络名称空间）。在这种情况下，将忽略 privnet 选项。

34.6 更多信息

LXC 容器驱动程序

<http://libvirt.org/drvlxc.html> 

35 从 LXC 迁移到 libvirt-lxc

从 SUSE Linux Enterprise Server 12 开始，LXC 已集成到 `libvirt` 库中。相比将 LXC 作为单独的解决方案使用，此决策可带来多项优势 — 例如，与其他虚拟化解决方案保持统一，或者在使用的内核方面获得独立性。本章将会说明需要执行哪些步骤来迁移可与 `libvirt` 库搭配使用的现有 LXC 环境。

35.1 主机迁移

迁移本身包括两个阶段。首先需要迁移主机，接着需迁移 LXC 容器。然后便可在 `libvirt` 环境中将原始容器作为 VM Guest 运行。

过程 35.1：主机迁移

1. 使用官方 DVD 媒体将主机升级到 SUSE Linux Enterprise Server 15。
2. 升级后，安装 `libvirt-daemon-lxc` 和 `libvirt-daemon-config-network` 软件包。
3. 基于现有的容器 `lxc_container` 创建 `libvirt` XML 配置 `lxc_container.xml`：

```
tux > sudo virt-lxc-convert /etc/lxc/lxc_container/config >
lxc_container.xml
```

4. 检查主机上的网络配置是否与容器配置文件中的配置相同，并根据需要予以修复。
5. 检查 `lxc_container.xml` 文件中是否存在任何异常或缺少的配置。请注意，某些 LXC 配置选项无法映射到 `libvirt` 配置。尽管转换通常应该会正常完成，但仍请查看第 34.4 节“`libvirt` LXC 驱动程序与 LXC 之间的差异”了解更多细节。
6. 根据创建的 XML 定义在 `libvirt` 中创建容器：

```
tux > sudo virsh -c lxc:/// define lxc_container.xml
```


35.2 容器迁移

迁移主机后，`libvirt` 中的 LXC 容器将无法引导。您需要将此容器也迁移到 SUSE Linux Enterprise Server 15 才能使一切正常运行。

过程 35.2：容器迁移

1. 缺少 `baseproduct` 文件（`zypper` 一直在指出此问题）。创建相关的符号链接：

```
root # R00TFS=/var/lib/lxc/lxc_container/rootfs
root # ln -s $R00TFS/etc/products.d/SUSE_SLES.prod $R00TFS/etc/products.d/
baseproduct
```

2. 添加 DVD 储存库。请注意，您需要将 DVD 设备替换为挂接到容器的相应设备：

```
root # zypper --root $R00TFS ar \
cd:///?devices=/dev/dvd SLES15-0
```

3. 禁用或删除以前的储存库：

```
root # zypper --root $R00TFS lr
| Alias                                | Name                                | Enabled |
Refresh
--+-+-----+-----+-----+-----+-----+-----+-----+
+-----+
1 | SLES12                                | SLES12                                | Yes     |
No
2 | SUSE-[...]-Server-12-SP3 38 | SUSE-[...]-Server-12-SP3 138 | Yes     |
No

root # zypper --root $R00TFS rr 2
```

4. 禁用或删除以前的储存库：

```
root # zypper --root $R00TFS lr
| Alias                                | Name                                | Enabled |
Refresh
--+-+-----+-----+-----+-----+-----+-----+
+-----+
```

1	openSUSE 42.3 Main	openSUSE 42.3 Main	Yes	
	No			
2	openSUSE 42.3 Update	openSUSE 42.3 Update	Yes	
	No			

```
root # zypper --root $ROOTFS rr 2
```

5. 升级容器：

```
root # zypper --root $ROOTFS dup
```

6. 安装极简模式，以确保安装所需的所有组件：

```
root # zypper --root $ROOTFS in -t pattern Minimal
```

35.3 启动容器

完成主机和容器迁移后，便能够启动容器：

```
root # virsh -c lxc:/// start lxc_container
```

如果您需要通过控制台查看容器生成的日志记录消息，请运行：

```
root # virsh -c lxc:/// console lxc_container
```

词汇表

一般

Dom0

该术语在 Xen 环境中使用，表示一个虚拟机。主机操作系统实际上是在特权域中运行的虚拟机，可称为 Dom0。主机上的所有其他虚拟机在非特权域中运行，可称为域 U。

KVM

请参见第 3 章 “KVM 虚拟化简介”

VHS

虚拟化主机服务器

运行 SUSE 虚拟化平台软件的物理计算机。虚拟化环境由超级管理程序、主机环境、虚拟机、关联的工具、命令和配置文件构成。其他常用术语包括主机、主机计算机 (Host Computer)、主机计算机 (Host Machine, HM)、虚拟服务器 (VS)、虚拟机主机 (VMH) 和 VM 主机服务器 (VHS)。

VirtFS

VirtFS 是全新的半虚拟化文件系统界面，旨在改进 KVM 环境中的直通方法。它以 VirtIO 框架为基础构建。

Xen

请参见第 2 章 “Xen 虚拟化简介”

xl

适用于 Xen 的一组命令，可让管理员通过主机计算机上的命令提示符管理虚拟机。它取代了已弃用的 xm 工具堆栈。

主机环境

允许与主机计算机环境交互的桌面或命令行环境。它提供命令行环境，并且还可包含 GNOME 或 IceWM 等图形桌面。主机环境作为特殊类型的虚拟机运行，拥有控制和管理其他虚拟机的特权。其他常用术语包括 Dom0、特权域和主机操作系统。

创建虚拟机向导

YaST 和虚拟机管理器中提供的一个软件程序，它提供图形界面来引导您完成创建虚拟机的步骤。该软件程序也可在文本模式下运行，在主机环境中的命令提示符处输入 **virt-install** 即可进入文本模式。

半虚拟化帧缓冲区

通过一个内存缓冲区（其中包含以半虚拟模式运行的虚拟机显示器的完整数据帧）驱动视频显示器的视频输出设备。

硬件辅助

Intel* 和 AMD* 提供虚拟化硬件辅助技术。此技术降低了 VM 输入/输出的频率（VM 陷阱更少），同时由于软件是主要开销来源，此技术也提高了效率（执行由硬件完成）。此外，此技术减少了内存占用量，可提供更好的资源控制，并可确保安全地指派特定的 I/O 设备。

虚拟化

在虚拟机上运行的 Guest 操作系统或应用程序。

虚拟机

能够托管 Guest 操作系统和关联的应用程序的虚拟化 PC 环境 (VM)，也可称为 VM Guest。

虚拟机管理器

一个软件程序，提供用于创建和管理虚拟机的图形用户界面。

超级管理程序

用于协调虚拟机与底层物理计算机硬件之间的低级交互的软件。

CPU

CPU 固定

使用 CPU 固定（也称为处理器亲和性）可在一个中央处理单元 (CPU) 或某个范围的 CPU 上绑定和取消绑定某个进程或线程。

CPU 热插拔

CPU 热插拔用于描述在不关闭系统的情况下更换/添加/拆除 CPU 的功能。

CPU 过量分配

利用虚拟 CPU 过量分配，可以为 VM 指派超出物理系统中实际存在的物理 CPU 数量的虚拟 CPU 数量。此过程并不会提高系统的总体性能，但在进行测试时可能有用。

CPU 限制

虚拟 CPU 限制允许您将 vCPU 容量设置为物理 CPU 容量的 1%–100%。

网络

传统网桥

主机为其提供了物理网络设备和虚拟网络设备的一种网桥。

内部网络

将虚拟机局限于其主机环境的一种网络配置。

外部网络

主机内部网络环境之外的网络。

无主机网桥

主机为其提供了物理网络设备但未提供虚拟网络设备的一种网桥。此网桥可让虚拟机在外部网络上通讯，但不能与主机通讯。这样，您便可以将虚拟机网络通讯与主机环境相隔离。

本地网桥

主机为其提供了虚拟网络设备但未提供物理网络设备的一种网桥。此网桥可让虚拟机与主机以及主机上的其他虚拟机通讯。虚拟机之间可以通过主机在外部网络上通讯。

桥接网络

这种网络连接允许在外部网络上将虚拟机标识为与其主机计算机相互独立且不相关的唯一身份。

空网桥

主机未为其提供任何物理网络设备或虚拟网络设备的一种网桥。此网桥可让虚拟机与同一主机上的其他虚拟机通讯，但不能与主机或在外部网络上通讯。

网络地址转换 (NAT)

允许虚拟机使用主机的 IP 地址和 MAC 地址的一种网络连接。

储存

AHCI

高级主机控制器接口 (AHCI) 是 Intel* 定义的一套技术标准，指定如何以不特定于实现的方式操作串行 ATA (SATA) 主机总线适配器。

xvda

分配给半虚拟计算机上的第一个虚拟磁盘的驱动器号。

原始磁盘

在单个字节级别而不是通过磁盘文件系统访问磁盘中的数据的一种方法。

块设备

以块的形式移动数据的数据储存设备，例如 CD-ROM 驱动器或磁盘驱动器。分区和卷也被视为块设备。

基于文件的虚拟磁盘

基于文件的虚拟磁盘，也称为磁盘映像文件。

稀疏映像文件

不保留其整个磁盘空间量，而是随着在其中写入数据而不断扩展的磁盘映像文件。

Linux 容器

cgroup

内核控制组（通常称为“cgroup”）是一项内核功能，可用于将任务（进程）及其子项聚合或划分成按层次结构组织的组，以隔离资源。

另请参见《系统分析和微调指南》，第 9 章“内核控制组”。

chroot

change root (chroot 或 change root jail) 是文件系统中的—个部分，它与操作系统的其他部分相隔离。为实现此目的，可以使用 **chroot** 或 **pivot_root** 命令来更改文件系统的根目录。在此类“chroot jail”中执行的程序无法访问指定目录树外部的文件。

内核名称空间

用于隔离—组进程的某些资源（例如网络、用户等）的内核功能。

容器

可将容器视为主机服务器上的一种“虚拟机”，它可以运行任何 Linux 系统，例如 openSUSE、SUSE Linux Enterprise Desktop 或 SUSE Linux Enterprise Server。与普通虚拟机之间的主要区别在于，容器将自己的内核与运行它的主机共享。

缩写词

ACPI

高级配置和电源接口 (ACPI) 规范为操作系统的设备配置和电源管理提供了开放的标准。

AER

高级错误报告

AER 是 PCI Express 规范提供的一项功能，用于报告 PCI 错误并在发生其中某些错误后进行恢复。

APIC

高级可编程中断控制器 (APIC) 是一个中断控制器系列。

BDF

总线:设备:功能

用于简要描述 PCI 和 PCIe 设备的表示法。

CG

控制组

用于限制、计算和隔离资源的使用（CPU、内存、磁盘 I/O 等）的功能。

EDF

最早截止时间优先

此调度程序以直观方式提供加权 CPU 共享，并使用实时算法来提供时间上的保障。

EPT

扩展页表

虚拟化环境中的性能与本机环境非常接近。不过，虚拟化确实会产生一定的开销。这些开销源自 CPU、MMU 和 I/O 设备的虚拟化。在一些新款 x86 处理器中，AMD 和 Intel 已开始提供硬件扩展来帮助弥补这种性能差距。2006 年，这两家供应商推出了采用 AMD-Virtualization (AMD-V) 和 Intel® VT-x 技术的第一代 x86 虚拟化硬件支持。最近，Intel 推出了其第二代硬件支持，其中整合了称作扩展页表 (EPT) 的 MMU 虚拟化。与使用影子分页技术进行 MMU 虚拟化相比，支持 EPT 的系统可以提升性能。如果工作负载不多，EPT 会增加内存访问延迟，这个代价可以通过在 Guest 和超级管理程序中有效使用大页来降低。

FLASK

Flux 高级安全内核

Xen 通过一个称为 FLASK 的安全体系结构，使用同名的模块来实现某种类型的强制访问控制。

HAP

高保证平台

HAP 结合了硬件和软件技术来提高工作站与网络安全性。

HVM

硬件虚拟机（Xen 通常这样称呼）。

IOMMU

输入/输出内存管理单元

IOMMU（AMD* 技术）是内存管理单元 (MMU)，可将支持直接内存访问 (DMA) 的 I/O 总线连接到主内存。

KSM

内核同页合并

KSM 可用于在 Guest 之间自动共享相同的内存页，以节省主机内存。KVM 经优化后可以使用 KSM（如果已在 VM 主机服务器上启用）。

MMU

内存管理单元

一个计算机硬件组件，负责处理 CPU 的内存访问请求。其功能包括虚拟地址到物理地址的转换（即虚拟内存管理）、内存保护、缓存控制、总线仲裁，在较为简单的计算机体系结构（尤其是 8 位系统）中，负责储存体切换。

PAE

物理地址扩展

32 位 x86 操作系统使用物理地址扩展 (PAE) 模式来实现 4 GB 以上物理内存的寻址。在 PAE 模式下，页表项 (PTE) 的大小为 64 位。

PCID

进程环境标识符

逻辑处理器可通过这些标识符缓存多个线性地址空间的信息，这样当软件切换到其他线性地址空间时，处理器能够保留缓存的信息。INVPCID 指令用于进行精细的 TLB 刷新，这对于内核会有所助益。

PCIe

高速外设组件互连

PCIe 用来替代旧式 PCI、PCI-X 和 AGP 总线标准。PCIe 融入了大量改进，包括更高的最大系统总线吞吐量、更低的 I/O 脚数，以及更小的物理占用空间。此外，它还提供更详细的错误检测和报告机制 (AER)，以及本机热插拔功能。它还向后兼容 PCI。

PSE 和 PSE36

扩展页大小

PSE 是指 x86 处理器的一项功能，允许页大于传统的 4 KiB 大小。PSE-36 功能在普通 10 位的基础上再额外提供 4 位，这些额外的位在指向大页的页目录项中使用。这样，便可以将大页放到 36 位地址空间中。

PT

页表

页表是计算机操作系统中的虚拟内存系统用来储存虚拟地址与物理地址之间的映射的数据结构。虚拟地址是访问进程特有的地址。物理地址是硬件 (RAM) 特有的地址。

QXL

QXL 是虚拟化环境的 cirrus VGA 帧缓冲 (8M) 驱动程序。

RVI 或 NPT

快速虚拟化索引、嵌套式页表

适用于处理器内存管理单元 (MMU) 的 AMD 第二代硬件辅助虚拟化技术。

SATA

串行 ATA

SATA 是一个计算机总线接口，可将主机总线适配器连接到硬盘和光驱等大容量储存设备。

SMEP

监督模式执行保护

用于防止 Xen 超级管理程序执行用户模式页，使通过应用程序恶意利用超级管理程序的许多企图更难以实现。

SPICE

适用于独立计算环境的简单协议

SXP

SXP 文件是 Xen 配置文件。

TCG

微代码生成器

模拟指令，而不是由 CPU 执行指令。

THP

透明大页

此功能可让 CPU 使用大于默认 4 KB 的页进行内存寻址。这有助于减少内存消耗量和 CPU 缓存使用率。KVM 经优化后可以通过 `madvise` 和机会性方法使用 THP（如果已在 VM 主机服务器上启用）。

TLB

转换后援缓冲区

TLB 是内存管理硬件用来提升虚拟地址转换速度的缓存。所有最新的台式机、笔记本电脑和服务端处理器都使用 TLB 来映射虚拟和物理地址空间，在任何使用虚拟内存的硬件中，几乎都会用到 TLB。

VCPU

一个调度实体，包含虚拟化 CPU 的每种状态。

VDI

虚拟桌面基础结构

VFIO

从内核 v3.6 开始，推出了一种从用户空间访问 PCI 设备的新方法，该方法称为 VFIO。

VHS

虚拟化主机服务器

VM root

VMM 以 VMX root 操作运行，Guest 软件以 VMX 非 root 操作运行。VMX root 操作与 VMX 非 root 操作之间的转换称为 VMX 转换。

VMCS

虚拟机控制结构

VMX 非 root 操作和 VMX 转换由一个称为虚拟机控制结构 (VMCS) 的数据结构来控制。VMCS 访问通过称为 VMCS 指针（每个逻辑处理器一个）的处理器状态组件来管理。VMCS 指针的值是 VMCS 的 64 位地址。通过 VMPTRST 和 VMPTRLD 指令来读取和写入 VMCS 指针。VMM 使用 VMREAD、VMWRITE 和 VMCLEAR 指令来配置 VMCS。VMM 可对它支持的每个虚拟机使用不同的 VMCS。如果虚拟机具有多个逻辑处理器（虚拟处理器），VMM 可对每个虚拟处理器使用不同的 VMCS。

VMQ

虚拟机设备队列

多队列网络适配器可在硬件级别支持多个 VM，它们能使不同的包队列关联到不同的托管 VM（通过 VM 的 IP 地址进行关联）。

VMM

虚拟机监视器（超级管理程序）

当处理器遇到与超级管理程序 (VMM) 相关的指令或事件时，它会退出 Guest 模式并回到 VMM。VMM 以远低于本机的速度模拟该指令或另一事件，然后返回到 Guest 模式。Guest 模式与 VMM 之间的来回转换属于高延迟操作，在此期间，Guest 执行会完全停滞。

VMX


虚拟机扩展

VPID

新的 TLB 软件控制支持（利用 VPID，您只需进行少量的 VMM 开发，就能提升 TLB 性能）。

VT-d

定向 I/O 虚拟化技术

类似于 Intel* 的 IOMMU (<https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices>) 。

vTPM

用于通过可信计算为 Guest 建立端到端完整性的组件。

基于 Seccomp2 的沙箱

为了增强恶意行为防范能力而只允许使用预先确定的系统调用的沙箱环境。

A 虚拟机驱动程序

利用虚拟化，您可以将工作负载整合到更新、更强大且更节能的硬件上。SUSE® Linux Enterprise Server 等半虚拟化操作系统及其他 Linux 发行套件能够感知底层虚拟化平台，因此可以有效地与它交互。Microsoft Windows* 等未经修改的操作系统无法感知虚拟化平台，需要直接与硬件交互。由于在整合服务器时无法做到这一点，因此必须为操作系统模拟硬件。模拟速度可能很慢，而且在模拟高吞吐量磁盘和网络子系统时尤为麻烦。大部分性能损失都是发生在这一环节。

SUSE Linux Enterprise 虚拟机驱动程序包 (VMDP) 包含适用于多种 Microsoft Windows 操作系统的 32 位和 64 位半虚拟化网络、总线与块驱动程序。这些驱动程序为未经修改的操作系统带来了半虚拟化操作系统具备的诸多性能优势：只有半虚拟化设备驱动程序能够感知到虚拟化平台（其他操作系统都不可以）。例如，对于操作系统而言，半虚拟化磁盘设备驱动程序就像是一个正常的物理磁盘，而设备驱动程序是直接与虚拟化平台交互的（无需模拟）。这有助于有效实现磁盘访问，使磁盘和网络子系统能够在虚拟化环境中以接近本机的速度运行，且无需对现有操作系统进行更改。


SUSE® Linux Enterprise 虚拟机驱动程序包作为 SUSE Linux Enterprise Server 的附加产品提供。有关详细信息，请参见 <https://www.suse.com/products/vmdriverpack/>。

有关详细信息，请参见官方的《VMDP Installation Guide》（VMDP 安装指南）（<https://documentation.suse.com/sle-vm dp/2.5/html/vmdp/index.html>）。

B 附录

B.1 安装半虚拟化驱动程序

B.1.1 安装适用于 Microsoft Windows* 的 virtio 驱动程序

SUSE 开发了适用于 Windows 的基于 virtio 的驱动程序，这些驱动程序包含在虚拟机驱动程序包 (VMDP) 中。有关 VMDP 的详细信息，请参见 <https://www.suse.com/products/vmdriverpack/> 。安装说明现在在专门的官方文档中提供。

C XM、XL 工具堆栈和 Libvirt 框架

C.1 Xen 工具堆栈

从早期发行版 Xen 2.x 开始，**xend** 就一直是事实上用于管理 Xen 安装的工具堆栈。Xen 4.1 中引入了一个处于技术预览状态的新工具堆栈 libxenlight（又称为 libxl）。libxl 是以 C 编写的小型低级别库，旨在为所有客户端工具堆栈（XAPI (<http://wiki.xen.org/wiki/XAPI>)、libvirt、xl）提供简单的 API。Xen 4.2 中将 libxl 提升为受官方支持状态，而将 **xend** 标记为弃用。Xen 4.3 和 4.4 系列中包含了 **xend**，以使用户有充足的时间将其工具过渡到 libxl。上游 Xen 项目中已去除 xend，从 Xen 4.5 系列和 SUSE Linux Enterprise Server 12 SP1 开始，将不再提供该工具堆栈。

尽管 SLES 11 SP3 包含了 Xen 4.2，但 SUSE 仍保留了 **xend** 工具堆栈，因为在服务包中进行这种有创性更改会给 SUSE Linux Enterprise 客户造成过大干扰。不过，SLES 12 将提供适当的机会让客户迁移到新的 libxl 工具堆栈，并去除已弃用且不再保留的 **xend** 堆栈。从 SUSE Linux Enterprise Server 12 SP1 开始，**xend** 将不再受支持。

xend 与 libxl 之间的主要差别之一是，前者是有状态的，而后者是无状态的。使用 **xend** 时，所有客户端应用程序（例如 **xm** 和 **libvirt**）都会看到相同的系统状态。**xend** 负责维护整个 Xen 主机的状态。在 libxl 中，**xl** 或 **libvirt** 等客户端应用程序必须维护状态。因此，使用 **xl** 创建的域对于 **libvirt** 等其他 libxl 应用程序是不可见或不可知的。一般情况下，我们建议不要混用多个 libxl 应用程序，而是使用单个 libxl 应用程序来管理 Xen 主机。在 SUSE Linux Enterprise Server 中，我们建议使用 **libvirt** 来管理 Xen 主机。这样，便可以通过 **virt-manager**、**virt-install**、**virt-viewer**、libguestfs 等 **libvirt** 应用程序来管理 Xen 系统。如果使用 **xl** 管理 Xen 主机，**libvirt** 将无法访问由 xl 管理的任何虚拟机。因此，任何 **libvirt** 应用程序也无法访问这些虚拟机。

C.1.1 从 xend/xm to xl/libxl 升级

xl 应用程序及其配置格式（请参见 **man xl.cfg**）可以向后兼容 **xm** 应用程序及其配置格式（请参见 **man xm.cfg**）。使用 **xl** 应该可以利用现有的 **xm** 配置。由于 libxl 是无状态的，并且 **xl** 不支持受管域的表示法，因此 SUSE 建议使用 **libvirt** 来管理 Xen 主机。SUSE

提供了一个名为 **xen2libvirt** 的工具，用于提供简单的机制来将以前由 **xend** 管理的域导入 **libvirt**。有关 **xen2libvirt** 的详细信息，请参见第 C.2 节 “将 Xen 域配置导入 **libvirt**”。

C.1.2 XL 设计

每个 **xl** 命令的基本结构如下：

```
xl subcommand OPTIONS DOMAIN
```

DOMAIN 是域 ID 编号或者域名（在内部转换为域 ID），**OPTIONS** 是特定于子命令的选项。

尽管 **xl/libxl** 可以向后兼容 **xm/xend**，但您应该注意两者之间的几项差别：

- 受管域或持久域。 **libvirt** 现在提供此项功能。
- **xl/libxl** 不支持在域配置文件中 **使用 Python 代码**。
- **xl/libxl** 不支持基于 SXP 格式配置文件创建域 (**xm create -F**)。
- **xl/libxl** 不支持通过域配置文件中的 **w!** 在 DomU 之间共享储存。

xl/libxl 相对较新且尚在大力开发之中，因此相比 **xm/xend** 工具堆栈仍然缺少一些功能。

- SCSI LUN/主机直通 (PVSCSI)
- USB 直通 (PVUSB)
- Xen 全虚拟化 Linux Guest 的直接内核引导

C.1.3 升级前的核对清单

在将 SLES 11 SP4 Xen 主机升级到 SLES 15 之前：

- 必须从 **xm** 域配置文件中去除任何 Python 代码。
- 建议使用 **virsh dumpxml DOMAIN_NAME DOMAIN_NAME.xml** 捕获所有现有虚拟机中的 **libvirt** 域 XML。
- 建议备份 **/etc/xen/xend-config.sxp** 和 **/boot/grub/menu.lst** 文件，以保留以前用于 Xen 的参数的参考信息。



注意

目前不支持将 SLES 11 SP4 Xen 主机上运行的虚拟机实时迁移到 SLES 15 Xen 主机。xend 与 libxl 工具堆栈两者的运行时环境不兼容。需要关闭虚拟机才能进行迁移。

C.2 将 Xen 域配置导入 libvirt

xen2libvirt 是用于将旧式 Xen 域配置导入 libvirt 虚拟化库的命令行工具。有关 libvirt 的详细信息，请参见《The Virtualization》（虚拟化）一书。使用 **xen2libvirt** 可以轻松将已弃用的 xm/xend 工具堆栈所管理的域导入新的 libvirt/libxl 工具堆栈中。使用此工具的 `--recursive` 模式可以一次性导入多个域

xen2libvirt 包含在 xen-tools 软件包中。如果需要，请使用以下命令安装该软件包

```
tux > sudo zypper install xen-tools
```

xen2libvirt 的一般语法为

```
xen2libvirt <options> /path/to/domain/config
```

其中 options 可以是：

-h、--help

列显有关 **xen2libvirt** 用法的简短信息。

-c、--convert-only

将域配置转换为 libvirt XML 格式，但不将配置导入 libvirt。

-r、--recursive

从指定的路径开始，以递归方式转换并/或导入所有域配置。

-f、--format

指定源域配置的格式。可以是 xm 或 sexpr（S 表达式格式）。

-v、--verbose

列显有关导入过程的更详细的信息。

例 C.1：将 XEN 域配置转换为 libvirt

假设您有一个通过 **xm** 管理的 Xen 域，`/etc/xen/sle12.xm` 中保存了该域的以下配置：

```
kernel = "/boot/vmlinuz-2.6-xenU"
memory = 128
name = "SLE12"
root = "/dev/hda1 ro"
disk = [ "file:/var/xen/sle12.img,hda1,w" ]
```

将此配置转换为 **libvirt** XML 而不导入，然后查看其内容：

```
tux > sudo xen2libvirt -f xm -c /etc/xen/sle12.xm > /etc/libvirt/qemu/sles12.xml
# cat /etc/libvirt/qemu/sles12.xml
<domain type='xen'>
  <name>SLE12</name>
  <uuid>43e1863c-8116-469c-a253-83d8be09aa1d</uuid>
  <memory unit='KiB'>131072</memory>
  <currentMemory unit='KiB'>131072</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='xenpv'>linux</type>
    <kernel>/boot/vmlinuz-2.6-xenU</kernel>
  </os>
  <clock offset='utc' adjustment='reset'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <disk type='file' device='disk'>
      <driver name='file'/>
      <source file='/var/xen/sle12.img'/>
      <target dev='hda1' bus='xen'/>
    </disk>
    <console type='pty'>
      <target type='xen' port='0'/>
    </console>
  </devices>
```

```
</domain>
```

要将域导入 `libvirt`，可以运行不带 `-c` 选项的相同 `xen2libvirt` 命令，或者使用导出的 `/etc/libvirt/qemu/sles12.xml` 文件并通过 `virsh` 定义新的 Xen 域：

```
tux > sudo virsh define /etc/libvirt/qemu/sles12.xml
```

C.3 `xm` 与 `xl` 应用程序之间的差异

本节罗列了 `xm` 与 `xl` 应用程序之间的所有差异。一般情况下，`xl` 可以与 `xm` 兼容。通常，用户只需在自定义脚本或工具中将 `xm` 替换为 `xl`。

您还可以通过 `virsh` 命令使用 `libvirt` 框架。本文档中只会显示 `virsh` 的第一个 `OPTION`。要获取有关此选项的更多帮助，请执行：

```
virsh help OPTION
```

C.3.1 表示法约定

为了让您轻松了解 `xl` 与 `xm` 命令之间的差异，本节使用了以下表示法：

表 C.1：表示法约定

表示法	含义
(-) minus	选项在 <code>xm</code> 中存在，但未包含在 <code>xl</code> 中。
(+) plus	选项在 <code>xl</code> 中存在，但未包含在 <code>xm</code> 中。

C.3.2 新的全局选项

表 C.2：新的全局选项

选项	任务
(+) <code>-v</code>	详细，提高输出的详细程度

选项	任务
(+) <u>-N</u>	试运行，不实际执行命令
(+) <u>-f</u>	强制执行。如果 xl 检测到 xend 也在运行，将会拒绝运行某些命令。此选项会强制执行这些命令，即使这样做不安全

C.3.3 未更改的选项

xl 和 **xm** 的常用选项列表，及其等效的 libvirt 选项。

表 C.3：通用选项

选项	任务	<u>libvirt</u> 等效选项
destroy <u>DOMAIN</u>	立即终止域。	virsh <u>destroy</u>
domid <u>DOMAIN_NAME</u>	将域名转换为 <u>DOMAIN_ID</u> 。	virsh <u>domid</u>
domname <u>DOMAIN_ID</u>	将 <u>DOMAIN_ID</u> 转换为 <u>DOMAIN_NAME</u> 。	virsh <u>domname</u>
help	显示简短的帮助消息（即常用命令）。	virsh <u>help</u>
pause <u>DOMAIN_ID</u>	暂停域。处于暂停状态的域仍会消耗分配的资源（例如内存），但不符合由 Xen 超级管理程序调度的条件。	virsh <u>suspend</u>
unpause <u>DOMAIN_ID</u>	使域脱离暂停状态。这样以前暂停的域便符合由 Xen 超级管理程序调度的条件。	virsh <u>resume</u>

选项	任务	libvirt 等效选项
<code>rename</code> <u>DOMAIN_ID</u> <u>NEW_DOMAIN_NAME</u>	将 <u>DOMAIN_ID</u> 的域名更改为 <u>NEW_DOMAIN_NAME</u> 。	<ol style="list-style-type: none"> 1. <pre>tux > virsh dumpxml DOMAINNAME > DOMXML</pre> 2. 修改 <u>DOMXML</u> 中的域名 3. <pre>tux > virsh undefine DOMAINNAME</pre> 4. <pre>tux > virsh define DOMAINNAME</pre>
<code>sysrq</code> <u>DOMAIN</u> <letter>	向域发送魔术系统请求，每种类型的请求由一个不同的字母表示。可以使用此选项向 Linux Guest 发送 SysRq 请求，有关详细信息，请参见 https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html 。需要在 Guest 操作系统中安装 PV 驱动程序。	<code>virsh send-keys</code> 只能针对 KVM 发送魔术系统请求
<code>vncviewer</code> <u>OPTIONS</u> <u>DOMAIN</u>	挂接到域的 VNC 服务器，并派生 <code>vncviewer</code> 进程。	<code>virt-viewer</code> <u>DOMAIN_ID</u> <code>virsh</code> <u>VNCDISPLAY</u>
<code>vcpu-set</code> <u>DOMAIN_ID</u> <u>VCPUS</u>	为相关的域设置虚拟 CPU 数量。与 <code>mem-set</code> 一样，此命令最多只能分配引导时为域配置的最大虚拟 CPU 数量。	<code>virsh setvcpus</code>

选项	任务	libvirt 等效选项
<code>vcpu-list</code> <u>DOMAIN_ID</u>	列出特定域的 VCPU 信息。如果未指定域，将提供所有域的 VCPU 信息。	<u>virsh</u> <u>vcpuinfo</u>
<code>vcpu-pin</code> <u>DOMAIN_ID</u> <VCPU all> <CPUs all>	固定 VCPU，使其仅在特定的 CPU 上运行。可以使用关键字 all 向域中的所有 VCPU 应用 CPU 列表。	<u>virsh</u> <u>vcupin</u>
<u>dmesg</u> [-c]	读取 Xen 消息缓冲区，与 Linux 系统上的 dmesg 类似。该缓冲区包含 Xen 引导过程中创建的信息性、警告和错误消息。	
<u>top</u>	执行 <u>xentop</u> 命令，该命令提供域的实时监视。 <u>xentop</u> 是一个 curses 接口。	<u>virsh</u> <u>nodecpustats</u> <u>virsh</u> <u>nodememstats</u>
<u>uptime</u> [-s] <u>DOMAIN</u>	列显正在运行的域的当前运行时间。使用 <u>xl</u> 命令时，必须指定 <u>DOMAIN</u> 参数。	
<u>debug-keys</u> <u>KEYS</u>	将调试键发送到 Xen。等同于按 Xen conswitch（默认为 Ctrl-A）三次，然后按“keys”。	
<u>cpupool-migrate</u> <u>DOMAIN</u> <u>CPU_POOL</u>	将 <u>DOMAIN_ID</u> 或 <u>DOMAIN</u> 指定的域移到 <u>CPU_POOL</u> 中。	

选项	任务	libvirt 等效选项
<u>cpupool-destroy</u> <u>CPU_POOL</u>	停用 CPU 池。仅当该 CPU 池中没有任何处于活动状态的域时，才可执行此操作。	
<u>block-detach</u> <u>DOMAIN_ID</u> <u>DevId</u>	分离域的虚拟块设备。 <u>devId</u> 可以是 Dom0 为设备指定的符号名称或设备数字 ID。需要运行 <u>xl block-list</u> 来确定该编号。	<u>virsh detach-disk</u>
<u>network-attach</u> <u>DOMAIN_ID</u> <u>NETWORK_DEVICE</u>	在 <u>DOMAIN_ID</u> 所指定的域中创建新网络设备。 <u>network_device</u> 描述要挂接的设备，使用与域配置文件中 vif 字符串相同的格式	<u>virsh attach-interface</u> <u>virsh attach-device</u>
<u>pci-attach</u> <u>DOMAIN</u> <BDF> [Virtual Slot]	将新的直通 PCI 设备热插入到指定的域。 BDF 是要直通的物理设备的 PCI 总线/设备/功能。	<u>virsh attach-device</u>
<u>pci-list</u> <u>DOMAIN_ID</u>	列出域的直通 PCI 设备	
<u>getenforce</u>	确定 FLASK 安全模块是否已装载并正在强制实施其策略。	
<u>setenforce</u> <1 0 Enforcing Permissive>	启用或禁用强制实施 FLASK 访问控制的功能。默认值为 permissive，您可以在超级管理程序的命令行上使用 flask_enforcing 选项更改默认值。	

C.3.4 已去除的选项

不再可用于 XL 工具堆栈的 `xm options` 列表，以及替代解决方法（如果有）。

C.3.4.1 域管理

已去除的域管理命令及其替代命令列表。

表 C.4：已去除的域管理选项

已去除的域管理选项		
选项	任务	等效选项
(-) <code>log</code>	列显 Xend 日志。	可在 <code>/var/log/xend.log</code> 中找到此日志文件
(-) <code>delete</code>	从 Xend 域管理中去除域。 <code>list</code> 选项显示域名。	<code>virsh undefine</code>
(-) <code>new</code>	将域添加到 Xend 域管理	<code>virsh define</code>
(-) <code>start</code>	启动使用 <code>xm new</code> 命令添加的 Xend 受管域	<code>virsh start</code>
(-) <code>dryrun</code>	试运行 - 列显 <code>SXP</code> 中生成的配置，但不创建域	<code>xl -N</code>
(-) <code>reset</code>	重设置域	<code>virsh reset</code>
(-) <code>domstate</code>	显示域状态	<code>virsh domstate</code>
(-) <code>serve</code>	通过 stdio 代理 Xend XMLRPC	
(-) <code>resume DOMAIN OPTIONS</code>	使域脱离挂起状态，并将其移回到内存中	<code>virsh resume</code>

已去除的域管理选项		
选项	任务	等效选项
(-) <u>suspend</u> <u>DOMAIN</u>	在状态文件中挂起域，以便稍后可以使用 <u>resume</u> 子命令将其恢复。与 <u>save</u> 子命令类似，但不能指定状态文件	<u>virsh</u> <u>managedsave</u> <u>virsh</u> <u>suspend</u>

C.3.4.2 USB 设备

USB options 不可用于 xl/libxl 工具堆栈。 virsh 提供 attach-device 和 detach-device 选项，但尚不适用于 USB。

表 C.5：已去除的 USB 设备管理选项

已去除的 USB 设备管理选项	
选项	任务
(-) <u>usb-add</u>	将新 USB 物理总线添加到域
(-) <u>usb-del</u>	从域中删除 USB 物理总线
(-) <u>usb-attach</u>	将新 USB 物理总线挂接到域的虚拟端口
(-) <u>usb-detach</u>	从域的虚拟端口分离 USB 物理总线
(-) <u>usb-list</u>	列出域的所有虚拟端口挂接状态
(-) <u>usb-list-assignable-devices</u>	列出所有可指派的 USB 设备
(-) <u>usb-hc-create</u>	创建域的新虚拟 USB 主机控制器
(-) <u>usb-hc-destroy</u>	销毁域的虚拟 USB 主机控制器

C.3.4.3 CPU 管理

CPU 管理选项发生了变化。我们提供了一些新选项，请参见：第 C.3.5.10 节 “`xl cpupool - *`”

表 C.6：已去除的 CPU 管理选项

已去除的 CPU 管理选项	
选项	任务
(-) <code>cpupool-new</code>	将 CPU 池添加到 Xend CPU 池管理
(-) <code>cpupool-start</code>	启动 Xend CPU 池
(-) <code>cpupool-delete</code>	从 Xend 管理中去除 CPU 池

C.3.4.4 其他选项

表 C.7：其他选项

其他已去除的选项	
选项	任务
(-) <code>shell</code>	启动交互式外壳
(-) <code>change-vnc-passwd</code>	更改 vnc 口令
(-) <code>vtpm-list</code>	列出虚拟 TPM 设备
(-) <code>block-configure</code>	更改块设备配置

C.3.5 已更改的选项

C.3.5.1 create

`xl create` CONFIG_FILE OPTIONS VARs



注意：libvirt 等效选项：

`virsh create`

表 C.8：已更改的 `xl create` 选项

已更改的 <code>create</code> 选项	
选项	任务
(*) <code>-f=FILE</code> 、 <code>--defconfig=FILE</code>	使用给定的配置文件

表 C.9：已去除的 `xm create` 选项

已去除的 <code>create</code> 选项	
选项	任务
(-) <code>-s</code> 、 <code>--skipdtd</code>	跳过 DTD 检查 - 创建之前跳过 XML 检查
(-) <code>-x</code> 、 <code>--xmldryrun</code>	XML 试运行
(-) <code>-F=FILE</code> 、 <code>--config=FILE</code>	使用给定的 SXP 格式配置脚本
(-) <code>--path</code>	在路径中搜索配置脚本
(-) <code>--help_config</code>	列显配置脚本的可用配置变量 (var)
(-) <code>-n</code> 、 <code>--dryrun</code>	试运行 — 列显 SXP 中的配置，但不创建域
(-) <code>-c</code> 、 <code>--console_autoconnect</code>	创建域后连接到控制台

已去除的 <code>create</code> 选项	
选项	任务
(-) <code>-q</code> 、 <code>--quiet</code>	安静模式
(-) <code>-p</code> 、 <code>--paused</code>	创建域后使其保持暂停状态

表 C.10：添加的 `xl create` 选项

添加的 <code>create</code> 选项	
选项	任务
(+) <code>-V</code> 、 <code>--vncviewer</code>	挂接到域的 VNC 服务器，并派生 <code>vncviewer</code> 进程
(+) <code>-A</code> 、 <code>--vncviewer-autopass</code>	通过 <code>stdin</code> 将 VNC 口令传递给 <code>vncviewer</code>

C.3.5.2 console

`xl console` `OPTIONS` `DOMAIN`



注意：libvirt 等效选项

`virsh console`

表 C.11：添加的 `xl console` 选项

添加的 <code>console</code> 选项	
选项	任务
(+) <code>-t</code> <code>[pv serial]</code>	连接到 PV 控制台，或连接到模拟的串行控制台。PV 域只能使用 PV 控制台，而 HVM 域可以使用上述两种控制台

C.3.5.3 info

`xl info`

表 C.12：已去除的 `xm info` 选项

已去除的 <code>info</code> 选项	
选项	任务
(-) <code>-n</code> 、 <code>--numa</code>	Numa 信息
(-) <code>-c</code> 、 <code>--config</code>	列出 Xend 配置参数

C.3.5.4 dump-core

`xl dump-core DOMAIN FILENAME`



注意：libvirt 等效选项

`virsh dump`

表 C.13：已去除的 `xm dump-core` 选项

已去除的 <code>dump-core</code> 选项	
选项	任务
(-) <code>-L</code> 、 <code>--live</code>	在不暂停域的情况下转储核心
(-) <code>-C</code> 、 <code>--crash</code>	转储核心后使域崩溃
(-) <code>-R</code> 、 <code>--reset</code>	转储核心后重设置域

C.3.5.5 list

`xl list options DOMAIN`



注意：libvirt 等效选项

virsh list --all

表 C.14：已去除的 `xm list` 选项

已去除的 <code>list</code> 选项	
选项	任务
(-) <code>-l</code> 、 <code>--long</code>	<code>xm list</code> 的输出以 <code>SXP</code> 格式呈现数据
(-) <code>--state==STATE</code>	输出处于指定状态的 VM 的信息

表 C.15：添加的 `xl list` 选项

添加的 <code>list</code> 选项	
选项	任务
(+) <code>-Z</code> 、 <code>--context</code>	同时列显安全标签
(+) <code>-v</code> 、 <code>--verbose</code>	同时列显域 UUID、关机原因和安全标签

C.3.5.6 `mem-*`



注意：libvirt 等效选项

virsh setmem

virsh setmaxmem

表 C.16：已更改的 `xl mem-*` 选项

已更改的 <code>mem-*</code> 选项	
选项	任务
<code>mem-max</code> <u>DOMAIN_ID</u> <u>MEM</u>	追加 <u>t</u> （表示 TB）、 <u>g</u> （表示 GB）、 <u>m</u> （表示 MB）、 <u>k</u> （表示 KB）和 <u>b</u> （表示字节）。指定域可以使用的最大内存量。
<code>mem-set</code> <u>DOMAIN_ID</u> <u>MEM</u>	使用气球驱动程序设置域的已用内存

C.3.5.7 `migrate`

`xl migrate` OPTIONS DOMAIN HOST



注意：libvirt 等效选项

`virsh migrate --live hvm-sles11-qcow2 xen+`
CONNECTOR://USER@IP_ADDRESS/

表 C.17：已去除的 `xm migrate` 选项

已去除的 <code>migrate</code> 选项	
选项	任务
(-) <u>-l</u> 、 <u>--live</u>	使用实时迁移。这会在不关闭域的情况下在主机之间迁移域
(-) <u>-r</u> 、 <u>--resource</u> <u>Mbs</u>	设置迁移域时允许达到的最大迁移速度 (Mbs)
(-) <u>-c</u> 、 <u>--change_home_server</u>	更改受管域的宿主服务器
(-) <u>--max_iters=</u> <u>MAX_ITERS</u>	在最终挂起之前的迭代次数（默认值为 30）

已去除的 <u>migrate</u> 选项	
选项	任务
(-) <u>--max_factor= MAX_FACTOR</u>	在最终挂起之前要传送的最大内存量（默认值为 3*RAM）。
(-) <u>--min_remaining= MIN_REMAINING</u>	在最终挂起之前的未写入页数（默认值为 50）
(-) <u>--abort_if_busy</u>	中止迁移，而不是执行最终挂起
(-) <u>--log_progress</u>	在 <u>xend.log</u> 中记录迁移进度
(-) <u>-s</u> 、 <u>--ssl</u>	使用 SSL 连接进行迁移

表 C.18：添加的 xl migrate 选项

添加的 <u>migrate</u> 选项	
选项	任务
(+) <u>-s SSHCOMMAND</u>	使用 <sshcommand> 而不是 <u>ssh</u>
(+) <u>-e</u>	在新主机上，不要在后台（在 <host> 上）等待域死机
(+) <u>-C CONFIG</u>	发送 <config> 而不是创建域时使用的配置文件

C.3.5.8 域管理

xl reboot OPTIONS DOMAIN



注意：libvirt 等效选项

virsh reboot

表 C.19：已去除的 `xm reboot` 选项

已去除的 <code>reboot</code> 选项	
选项	任务
(-) <code>-a</code> 、 <code>--all</code>	重引导所有域
(-) <code>-w</code> 、 <code>--wait</code>	等待重引导完成后再返回。这可能需要一段时间，因为需要干净地关闭域中的所有服务

表 C.20：添加的 `xl reboot` 选项

添加的 <code>reboot</code> 选项	
选项	任务
(+) <code>-F</code>	对于没有 PV 驱动程序的 HVM Guest，回退到 ACPI 重设置事件

`xl save` `OPTIONS` `DOMAIN` `CHECK_POINT_FILE` `CONFIG_FILE`



注意：libvirt 等效选项

`virsh save`

表 C.21：添加的 `xl save` 选项

添加的 <code>save</code> 选项	
选项	任务
(+) <code>-c</code>	创建快照后使域保持运行状态

`xl restore` `OPTIONS` `CONFIG_FILE` `CHECK_POINT_FILE`



注意：libvirt 等效选项

`virsh restore`

表 C.22：添加的 xl restore 选项

添加的 restore 选项	
选项	任务
(+) <u>-p</u>	恢复域后不将其取消暂停
(+) <u>-e</u>	在新主机上不在后台等待域死机
(+) <u>-d</u>	启用调试消息
(+) <u>-V</u> 、 <u>--vncviewer</u>	挂接到域的 VNC 服务器，并派生 vncviewer 进程
(+) <u>-A</u> 、 <u>--vncviewer-autopass</u>	通过 stdin 将 VNC 口令传递给 vncviewer

xl shutdown OPTIONS DOMAIN



注意：libvirt 等效选项

virsh shutdown

表 C.23：已去除的 xm shutdown 选项

已去除的 shutdown 选项	
选项	任务
(-) <u>-w</u> 、 <u>--wait</u>	等待域完成关机后再返回
(-) <u>-a</u>	关闭所有 Guest 域
(-) <u>-R</u>	
(-) <u>-H</u>	

表 C.24：添加的 xl shutdown 选项


添加的 shutdown 选项	
选项	任务
(+) <u>-F</u>	如果 Guest 不支持 PV 关机控制，则回退为发送 ACPI 电源事件

表 C.25：已更改的 xl trigger 选项

已更改的 trigger 选项	
选项	任务
<u>- trigger DOMAIN <nmi reset init power sleep s3resume> VCPU</u>	向域发送触发器。仅适用于 HVM 域

C.3.5.9 xl sched-*

xl sched-credit OPTIONS

 注意：libvirt 等效选项

virsh schedinfo

表 C.26：已去除的 xm sched-credit 选项

已去除的 sched-credit 选项	
选项	任务
<u>-d DOMAIN、--domain= DOMAIN</u>	域
<u>-w WEIGHT、--weight= WEIGHT</u>	权重为 512 的域将获得的 CPU 是所争用的主机上权重为 256 的域的两倍。合法权重范围为 1 到 65535，默认值为 256

已去除的 <code>sched-credit</code> 选项	
选项	任务
<code>-c CAP</code> 、 <code>--cap= CAP</code>	CAP 可选择性修复域能够消耗的最大 CPU 数量

表 C.27：添加的 `xl sched-credit` 选项

添加的 <code>sched-credit</code> 选项	
选项	任务
(+) <code>-p CPUP00L</code> 、 <code>--cpupool= CPUP00L</code>	将输出内容限制为指定 CPU 池中的域
(+) <code>-s</code> 、 <code>--schedparam</code>	指定此选项可列出或设置池范围的调度程序参数
(+) <code>-t TSLICE</code> 、 <code>--tslice_ms= TSLICE</code>	时间片 (TSLICE) 告知调度程序要允许 VM 运行多长时间后再开始抢占模式
(+) <code>-r RLIMIT</code> 、 <code>--ratelimit_us= RLIMIT</code>	Ratelimit 会尝试限制每秒调度次数

`xl sched-credit2 OPTIONS`



注意：libvirt 状态

`virsh` 仅支持 `credit` 调度程序，不支持 `credit2` 调度程序

表 C.28：已去除的 `xm sched-credit2` 选项

已去除的 <code>sched-credit2</code> 选项	
选项	任务
<code>-d DOMAIN</code> 、 <code>--domain= DOMAIN</code>	域

已去除的 <code>sched-credit2</code> 选项	
选项	任务
<code>-w WEIGHT</code> 、 <code>--weight= WEIGHT</code>	合法权重范围为 1 到 65535，默认值为 256

表 C.29：添加的 `x1 sched-credit2` 选项

添加的 <code>sched-credit2</code> 选项	
选项	任务
<code>(+) -p CPUP00L</code> 、 <code>--cpupool= CPUP00L</code>	将输出内容限制为指定 CPU 池中的域

`x1 sched-sedf OPTIONS`

表 C.30：已去除的 `xm sched-sedf` 选项

已去除的 <code>sched-sedf</code> 选项	
选项	任务
<code>-p PERIOD</code> 、 <code>--period= PERIOD</code>	常规 <code>EDF</code> 调度用法，以毫秒为单位
<code>-s SLICE</code> 、 <code>--slice= SLICE</code>	常规 <code>EDF</code> 调度用法，以毫秒为单位
<code>-l LATENCY</code> 、 <code>--latency= LATENCY</code>	域执行大量 I/O 时延长的时段
<code>-e EXTRA</code> 、 <code>--extra= EXTRA</code>	允许域额外运行一段时间的标志（0 或 1）
<code>-w WEIGHT</code> 、 <code>--weight= WEIGHT</code>	另一种设置 CPU 片的方式

表 C.31：添加的 `x1 sched-sedf` 选项

添加的 <code>sched-sedf</code> 选项	
选项	任务
<code>(+) -c CPUP00L</code> 、 <code>--cpupool= CPUP00L</code>	将输出内容限制为指定 CPU 池中的域
<code>(+) -d DOMAIN</code> 、 <code>--domain= DOMAIN</code>	域

C.3.5.10 **xl cpupool -***

xl cpupool-cpu-remove CPU_P00L <CPU nr>|node:<node nr>

xl cpupool-list [-c|--cpus] CPU_P00L

表 C.32：已去除的 **xl cpupool-list** 选项

已去除的 cpupool-* 选项	
选项	任务
(-) <u>-l</u> 、 <u>--long</u>	以 SXP 格式输出所有 CPU 池细节

xl cpupool-cpu-add CPU_P00L cpu-nr|node:node-nr

xl cpupool-create OPTIONS CONFIG_FILE [Variable=Value ...]

表 C.33：已去除的 **xl cpupool-create** 选项

已去除的 cpupool-create 选项	
选项	任务
(-) <u>-f</u> <u>FILE</u> 、 <u>--defconfig=</u> <u>FILE</u>	使用给定的 Python 配置脚本。处理参数后会装载该配置文件
(-) <u>-n</u> 、 <u>--dryrun</u>	试运行 - 列显 SXP 中生成的配置，但不创建 CPU 池
(-) <u>--help_config</u>	列显配置脚本的可用配置变量 (var)
(-) <u>--path=</u> <u>PATH</u>	在路径中搜索配置脚本。PATH 的值是冒号分隔的目录列表
(-) <u>-F=</u> <u>FILE</u> 、 <u>--config=</u> <u>FILE</u>	要使用的 CPU 池配置 (SXP)

C.3.5.11 **PCI 和块设备**

xl pci-detach [-f] DOMAIN_ID <BDF>



注意：libvirt 等效选项

virsh detach-device

表 C.34：添加的 **xl pci-detach** 选项

添加的 <u>pci-detach</u> 选项	
选项	任务
(+) <u>-f</u>	如果指定了 <u>-f</u> ，即使在没有 Guest 协作的情况下， xl 也会强制去除设备

表 C.35：已去除的 **xm block-list** 选项

已去除的 <u>block-list</u> 选项	
选项	任务
(-) <u>-l</u> 、 <u>--long</u>	列出域的虚拟块设备

表 C.36：其他选项

选项	<u>libvirt</u> 等效选项
xl <u>block-attach</u> <u>DOMAIN</u> <disk-spec-component(s)>	<u>virsh</u> <u>attach-disk/attach-device</u>
xl <u>block-list</u> <u>DOMAIN_ID</u>	<u>virsh</u> <u>domblklist</u>

C.3.5.12 网络

表 C.37：网络选项

选项	<u>libvirt</u> 等效选项
xl <u>network-list</u> <u>DOMAIN(s)</u>	<u>virsh</u> <u>domiflist</u>

选项	libvirt 等效选项
<code>xl network-detach DOMAIN_ID devid mac</code>	<code>virsh detach-interface</code>
<code>xl network-attach DOMAIN(s)</code>	<code>virsh attach-interface/attach-device</code>

表 C.38：已去除的 `xl network-attach` 选项

已去除的选项	
选项	任务
<code>(-) -l、--long</code>	

C.3.6 新选项

表 C.39：新选项

选项	任务
<code>config-update DOMAIN CONFIG_FILE OPTIONS VARS</code>	更新为运行中的域保存的配置。此更新不会立即生效，而是在 Guest 下次重新启动时应用。此命令可用于确保在 Guest 重新启动时保留对 Guest 进行的运行时修改
<code>migrate-receive</code>	
<code>sharing DOMAIN</code>	列出共享页的计数。专门列出指定的域的该信息。如果未指定域，则列出所有域的该信息
<code>vm-list</code>	列显有关 Guest 的信息。此列表不包括有关 Dom0 和存根域等服务或辅助域的信息
<code>cpupool-rename CPU_POOL NEWNAME</code>	将 CPU 池重命名为 newname

选项	任务
<code>cpupool-numa-split</code>	将计算机分割为在每个 numa 节点一个 CPU 池
<code>cd-insert DOMAIN <VirtualDevice> <type:path></code>	将 CD-ROM 插入 Guest 域的现有虚拟 CD 驱动器。该虚拟驱动器必须已存在，但当前可以是空的
<code>cd-eject DOMAIN <VirtualDevice></code>	从 Guest 的虚拟 CD 驱动器中弹出 CD-ROM。仅适用于 HVM 域
<code>pci-assignable-list</code>	列出所有可指派的 PCI 设备。它们是系统中配置为可直通且已绑定到 Dom0 中的适当 PCI 后端驱动程序（而不是真实驱动程序）的设备
<code>pci-assignable-add <BDF></code>	使 PCI 总线/设备/功能 BDF 中的设备可指派到 Guest。这会将该设备绑定到 pciback 驱动程序
<code>pci-assignable-remove OPTIONS <BDF></code>	使 PCI 总线/设备/功能 BDF 中的设备可指派到 Guest。这至少会从 pciback 取消绑定该设备
<code>loadpolicy POLICY_FILE</code>	从给定的策略文件装载 FLASK 策略。初始策略将以多重引导模块的形式提供给超级管理程序；此命令允许对策略进行运行时更新。装载新安全策略会重置对设备标签进行的运行时更改

C.4 外部链接

有关 Xen 工具堆栈的详细信息，请参见以下联机资源：

Xen 中的 XL

XL in Xen 4.2 (Xen 4.2 中的 XL) (http://wiki.xenproject.org/wiki/XL_in_Xen_4.2) ↗

xl 命令

XL (<http://xenbits.xen.org/docs/unstable/man/xl.1.html>)  命令行。

xl.cfg

xl.cfg (<http://xenbits.xen.org/docs/unstable/man/xl.cfg.5.html>)  域配置文件语法。

xl disk

xl disk (<https://xenbits.xen.org/docs/4.3-testing/misc/xl-disk-configuration.txt>)  配置选项。

XL 与 Xend

XL vs Xend (XL 与 Xend) (http://wiki.xenproject.org/wiki/XL_vs_Xend_Feature_Comparison)  功能比较。

BDF 文档

BDF 文档 (http://wiki.xen.org/wiki/Bus:Device.Function_%28BDF%29_Notation) .

libvirt

virsh (<http://libvirt.org/virshcmdref.html>)  命令。

C.5 以 **xm** 兼容的格式保存 Xen Guest 配置

尽管 **xl** 是当前用于管理 Xen Guest 的工具集（此外还有首选的 **libvirt**），但您可能需要将 Guest 配置导出为过去使用的 **xm** 格式。要实现此目的，请执行以下步骤：

1. 首先将 Guest 配置导出到某个文件中：

```
tux > virsh dumpxml guest_id > guest_cfg.xml
```

2. 然后将配置转换为 **xm** 格式：

```
tux > virsh domxml-to-native xen-xm guest_cfg.xml > guest_xm_cfg
```

D GNU 许可证

本附录包含 GNU 自由文档许可证版本 1.2。

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. Secondly, 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 <http://www.gnu.org/copyleft/7>.

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.