虚拟化 --- KVM （ kernel-based VM ）

迁移：

系统的迁移是指把源主机上的操作系统和应用程序移动到目的主机，并且能够在目的主机上正常运行。在没有虚拟机的时代，物理机之间的迁移依靠的是系统备份和恢复技术。在源主机上实时备份操作系统和应用程序的状态，然后把存储介质连接到目标主机上，最后在目标主机上恢复系统。随着虚拟机技术的发展，系统的迁移更加灵活和多样化

虚拟化 --- KVM （ kernel-based VM ）

KVM 虚拟机的热迁移 ---Live Migration

服务器虚拟化技术是当前的热点，而虚拟机的“热迁移（ Live Migration ）”技术则是虚拟化技术当中的热点

热迁移（又叫动态迁移、实时迁移），即虚拟机保存（ save ） / 恢复 (restore) ：将整个虚拟机的运行状态完整保存下来，同时可以快速的恢复到原有硬件平台甚至是不同硬件平台上。恢复以后，虚拟机仍旧平滑运行，用户不会察觉到任何差异。

虚拟化 --- KVM （ kernel-based VM ）

迁移的种类：

P2P ：物理机之间的迁移
V2P ：虚拟机迁到物理机
P2V ：物理机迁到虚拟机
V2V ：虚拟机迁到虚拟机

虚拟化 --- KVM （ kernel-based VM ）

热迁移的应用

1. 虚拟机的热迁移技术最初是被用于双机容错或者负载均衡：当宿主机出现软硬件故障导致服务异常时，虚拟机可漂移到另外主机上，或者在集群中依据工作负载量的大小，选择更换宿主机与否来保证自身良好的服务提供性

虚拟化 --- KVM （ kernel-based VM ）

热迁移的应用

2. 系统硬件维护：当前很多操作系统都能够稳定支持 7×24 运行，但是硬件却需要定期的进行维护。如果使用虚拟机的动态迁移技术，将虚拟机从需要维护的物理机器迁移到另外一主器，等维护完成后，在将其迁回到原来的物理机器。所有的系统服务和应用程序在迁移 & 恢复后仍旧正常运行，用户不会察觉到由于硬件维护造成的中断。最终实现了我们服务不受硬件维护干扰的 7*24 小时的工作愿望

虚拟化 --- KVM （ kernel-based VM ）

热迁移的应用

3. 数据库备份：对于一些大型、关键的数据库应用，备份是一项重要但复杂的工作。虚拟机的保存 / 恢复可以将数据库运行在虚拟机中，如需备份就保存虚拟机，这样数据库中的所有数据、状态都做了备份。如果数据库崩溃了，就可以通过恢复虚拟机来恢复整个数据库

虚拟化 --- KVM （ kernel-based VM ）

热迁移的应用

4. 环境重现：进行性能测试或程序调试时，都需要重现当时复杂庞大并且与实时完全一致的网络环境，不仅仅是重启、配置软件，而且常常需要一定的运行时间。我们可以将各服务安装到独立的各个虚拟机，然后利用各个独立虚拟机部署我们所需的工作环境，可以大大缩短环境重现时间

虚拟化 --- KVM （ kernel-based VM ）

热迁移的应用

5. 计算机共享： 在一些公共场合用户需要共享计算机，但是由于不同的系统配置和软件需要花费大量的时间来配置和恢复。这时使用虚拟机的保存 / 恢复可以很好地解决这个问题。或者也可以，在物理机上运行多个逻辑虚拟机帮助我们分配给每一个需要者一个独立，安全，稳定的环境。当因为宿主机出现问题影响虚拟机使用时，我们可以让该虚拟机漂移到其他宿主机以此来保持正常工作

虚拟化 --- KVM （ kernel-based VM ）

最终我们迁移的目的就是：
简化系统维护管理
高系统负载均衡
增强系统错误容忍度
优化系统电源管理

虚拟化 --- KVM （ kernel-based VM ）

热迁移的优势

1. 首先是可伸缩性比较强， IT 管理者可以在合理时间段让运行某些关键业务的服务器适当减少工作量，以便进行更新操作系统，给应用程序打补丁等。而到了服务高峰期，又可以弹性地进行大负载量的运算。

虚拟机迁移过程完全透明，几乎不影响使用

虚拟化 --- KVM （ kernel-based VM ）

热迁移的优势

2. 其次，现在的数据中心都追求环保节能， 工作量负载大的应用程序必然会令服务器能耗增加，有了虚拟机热迁移技术，当一台物理服务器负载过大时，系统管理员可以将其上面的虚拟机迁移到其他服务器，可有效减低数据中心服务器的总体能耗，再通过冷却系统将数据中心的温度保持在正常水平

虚拟化 --- KVM （ kernel-based VM ）

热迁移的局限：

但是，进行虚拟机的热迁移也有不少的限制。例如， VMotion 在进行迁移之前，管理软件会检测目标服务器的 X86 架构是否与原服务器兼容。包括存储设备以及处理器，虚拟机必须放到共享的存储里， CPU 的类型也要一样，不仅不能一个是英特尔，一个是 AMD ，甚至相同厂商不同产品线的 CPU 也不行，比如英特尔至强和奔腾

虚拟化 --- KVM （ kernel-based VM ）

我们可以从以下几个方面衡量虚拟机迁移的效率

虚拟化 --- KVM （ kernel-based VM ）

1. 整体迁移时间：从源主机中迁移操作开始到目的主机上客户机服务处于不可用状态的时间，此时源主机上客户机已经暂停服务，目的主机上的客户机还未恢复服务

虚拟化 --- KVM （ kernel-based VM ）

2. 服务器停机时间：在迁移过程中，源主机和目的主机上的客户机都处于不可用状态的时间，此时源主机上客户机已暂停，目的目的主机上客户还未恢复服务

虚拟化 --- KVM （ kernel-based VM ）

3. 对服务的性能影响：不仅包括迁移后的客户机中应用程序的性能与迁移前相对比是否有所降低，还包括迁移后对目的主机上的其他服务的性能影响

虚拟化 --- KVM （ kernel-based VM ）

Kvm 动态迁移，也有如下几个建议和注意事项

虚拟化 --- KVM （ kernel-based VM ）

1. 源宿主机和目的宿主机直接尽量用网络共享的存储系统来保存客户机磁盘镜像。
例如 NFS ， ISCSI ， Glusterfs 等

虚拟化 --- KVM （ kernel-based VM ）

2. 为了提高动态迁移的成功率，尽量在同类型 cpu 的主机上面进行动态迁移，尽管 kvm 动态迁移也支持从 Intel 平台迁移到 amd 平台。
但，从安全性，稳定度考虑不建议这样去操作！！！

虚拟化 --- KVM （ kernel-based VM ）

3.64 位的客户机只能运行在 64 宿主机之间的迁移，而 32 位客户机可以在 32 宿主机和 64 位宿主机之间迁移

虚拟化 --- KVM （ kernel-based VM ）

4. 动态迁移的源宿主机和目的宿主机对 NX 位的设置是相同，要么同为关闭状态，要么同为打开状态。在 Intel 平台上的 linux 系统中，用“ cat /proc/couinfo |grep nx ” 命令可以查看是否有 NX 的支持

虚拟化 --- KVM （ kernel-based VM ）

NX ，全名为“ No eXecute” ，即“禁止运行”，是应用在 CPU 的一种技术，用作把存储器区域分隔为只供存储处理器指令集，或只供数据使用。任何使用 NX 技术的存储器，代表仅供数据使用，因此处理器的指令集并不能在这些区域存储。这种技术可防止大多数的缓冲溢出攻击，即一些恶意程序，把自身的恶意指令集放在其他程序的数据存储区并运行，从而把整台计算机控制

虚拟化 --- KVM （ kernel-based VM ）

5. 在进行动态迁移时，被迁移客户机的名称是唯一的，在目的宿主机上不能有与源宿主机被迁移客户机同名的客户机存在

虚拟化 --- KVM （ kernel-based VM ）

6. 目的宿主机和源宿主机的软件尽可能的相同。 也就是同为 Vmware ， KVM ， Xen 等

虚拟化 --- KVM （ kernel-based VM ）

安装前准备工作：

迁移需要识别 source&target 主机的主机名
所以需要我们搭建 DNS Ser 或者将 IP& 域名 & 主机名的对应条目写入到 hosts 文件中去

虚拟化 --- KVM （ kernel-based VM ）

实验环境：

Source Ser ---- VM
Target Ser
NFS Server 共享存储服务器

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：
NFS Server

1. 安装 / 配置 / 开启 NFS 服务
共享 NFS Server 上的目录： /migration
2. 编辑 hosts 文件

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：
Source Server

1. 挂载 NFS Ser 的 /migration 目录到本地的 /var/lib/libvirt/images 目录上
2. 安装 VM
3. 编辑 hosts 文件

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：
Target Server

1. 挂载 NFS Ser 的 /migration 目录到本地的 /var/lib/libvirt/images 目录上
2. 编辑 hosts 文件

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：

完成以上实验环境搭建，我们来 Live Migration

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：

完成以上实验环境搭建，我们来 Live Migration

1. 分别在两台机器上查看当前运行在本地的虚拟机
Virsh list –all
确定将 source 上的 vm 挂载到 target ，并且确实在 target host 上没有同名的主机

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：

完成以上实验环境搭建，我们来 Live Migration

2.source host
#Virsh migrate --live VMname qemu+ssh://Target-IP/system
#virsh list –all
查看本地运行的虚拟机来验证是否迁移成功

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：

完成以上实验环境搭建，我们来 Live Migration

3.Target Host
#virsh list –all
查看 Target Host 上运行的虚拟机，验证是否迁移成功

虚拟化 --- KVM （ kernel-based VM ）

实验步骤：

当然，迁移工作也可以在我们的 V-manager 上得以实现。