虚拟机在线迁移技术


一张图带你了解虚拟机在线迁移技术
服务器虚拟化一经出现就被视为IT界的Game-Changing技术
虚拟化的本质是将服务器上的硬件资源经过虚拟化技术变成一个可以拆分的资源，
拆分后的资源也就是一个个虚拟主机，都拥有完整的原有硬件可提供的服务能力。
其带来的好处和变革，对传统IDC影响巨大:
故障隔离
节能环保
提高资源
利用率
故障快速
恢复
那么，在提高物理服务器资源利用率的同时
如何避免对该物理服务器上虚拟机的影响呢 ?
1
分布式服务
我们可以将物理机的存储、内存作为分布式的服务提供出去
减少对单机容量的依赖，UCloud有对应的产品可供使用，
例如UDisk对应存储，UMem对应内存，等等.....
2
在线迁移
虚拟化技术底层也提供了一个杀手级的方案，那就是我们
今天要介绍的“虚拟机在线迁移技术”
在线迁移技术，让虚拟机的可用率>物理机可用率成为了可能
什么是虚拟机的在线迁移技术？
所谓在线迁移，也就是可以在不停机的情况下，
将虚拟机从一台物理机迁移到另外一台物理机，
迁移过程不影响虚拟机的正常运行，
也就不会影响用户跑在虚拟机上的业务。
这个特性，让虚拟机的可用率>物理机可用率成为了可能

UCloud 利用在线迁移技术，提高虚拟机可用性的使用场景

物理机故障，提前迁移

很多时候，物理机宕机或者故障是有先兆的，例如内核会报硬件错误，在收到这个告警之后和物理宕机之前这段时间内，我们通过在线迁移将该物理机上的虚拟机迁移到一台状态健康物理机上，避免虚拟机受到物理机宕机影响，保障用户业务的正常。

Raid卡故障，在线迁移

物理机raid卡故障，这个尽管不会导致宕机，但一旦出现就造成虚拟机的IO急剧下降，相当于虚拟机不可用，我们同样通过在线迁移将虚拟机迁移走，保证虚拟机的IO性能。

避免受二次宕机影响

物理机在毫无前兆的情况宕机了，那重复宕机的可能性就很大。为了避免该物理机上用户虚拟机受到二次宕机的影响，在物理机恢复后，我们马上迁移走该物理机上面的虚拟机，将物理机下架检查修复。

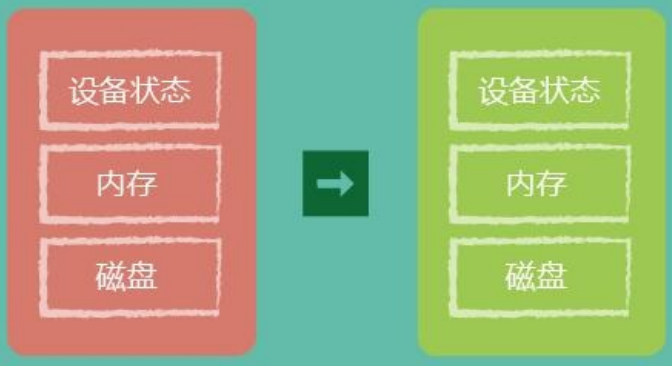
通过在线迁移，降低物理机负载

当物理机的负载到达瓶颈，那么上面的虚拟机性能也会受到影响，我们通过监控发现负载问题，通过在线迁移上面的部分虚拟机，来降低物理机负载，以达到资源利用率和虚拟机性能的平衡。

深入剖析：虚拟机在线迁移技术是如何实现的？

在线迁移是如何做到在几乎不停机的情况下，
将一台虚拟机完整的迁移到另外一台物理服务上的呢？

虚拟机的数据分为三类， 磁盘，内存，设备状态（例如虚拟CPU寄存器）
在线迁移就是把这三类数据在保证一致的情况下迁移到另外一台宿主机



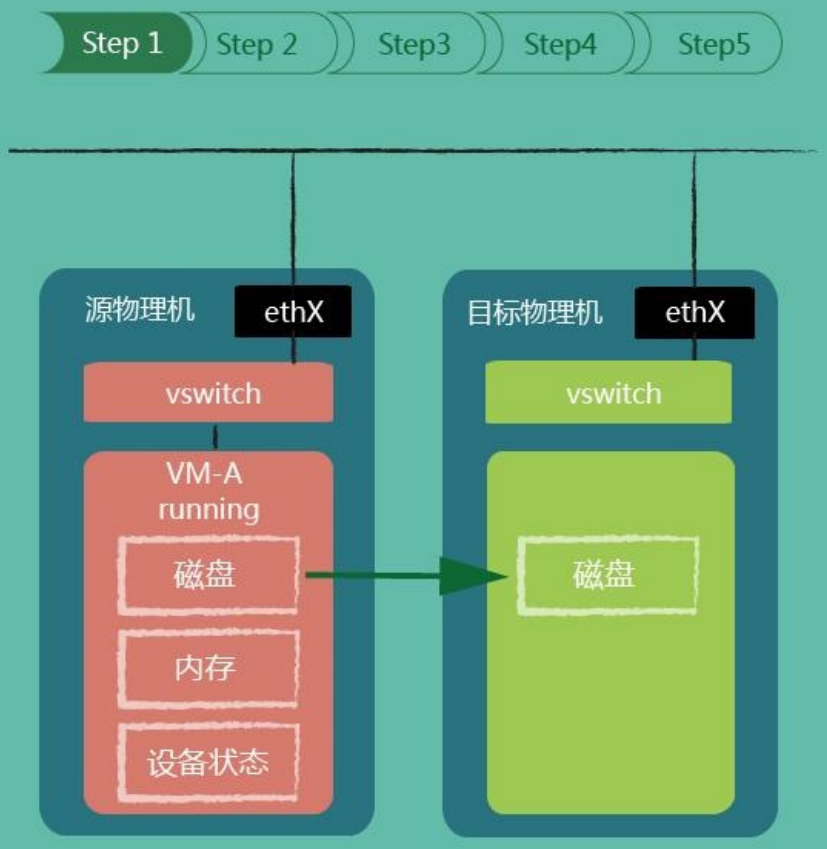
这里我们以基于KVM虚拟化方案（底层为KVM，模拟器为qemu，管理器为libvirt）的在线迁移为例，说明在线迁移是如何达成的

KVM的在线迁移分为两类：

这里重点说下第二类，因为第一类是第二类的一个特例。
第二类也有若干种细分，这里只讨论带块设备的完整迁移的方案。

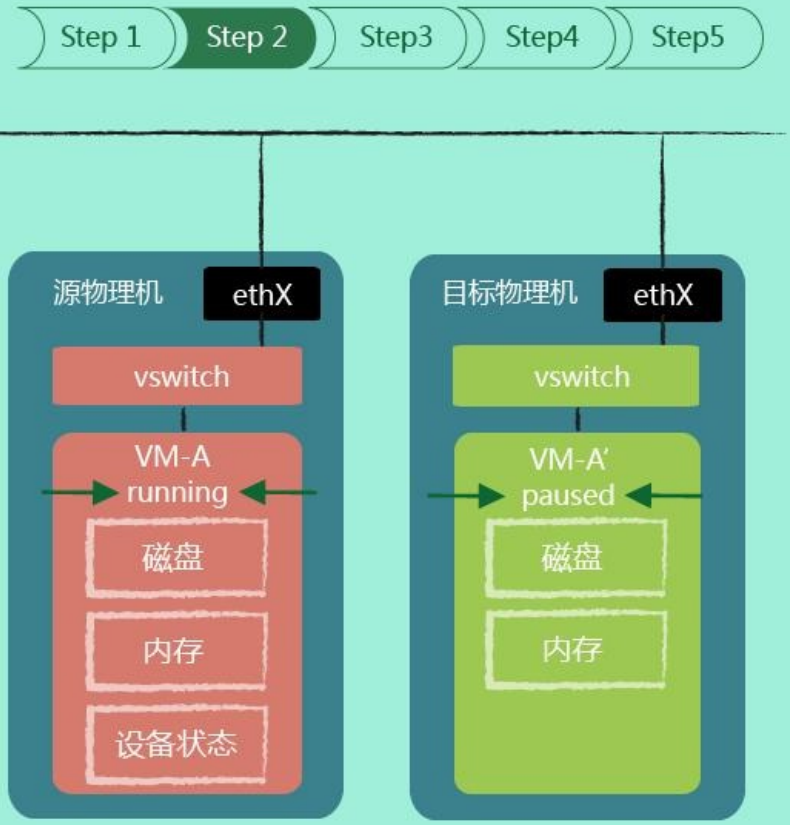
数据的迁移基于tcp协议通过网络来传输完成的，
因此迁移之前必须保证两台物理机在网络上是必须通的。

假设将从将源物理机的一个虚拟机VM-A
迁移到目标物理机VM-A’ ，带块设备的在线迁移过程如下：

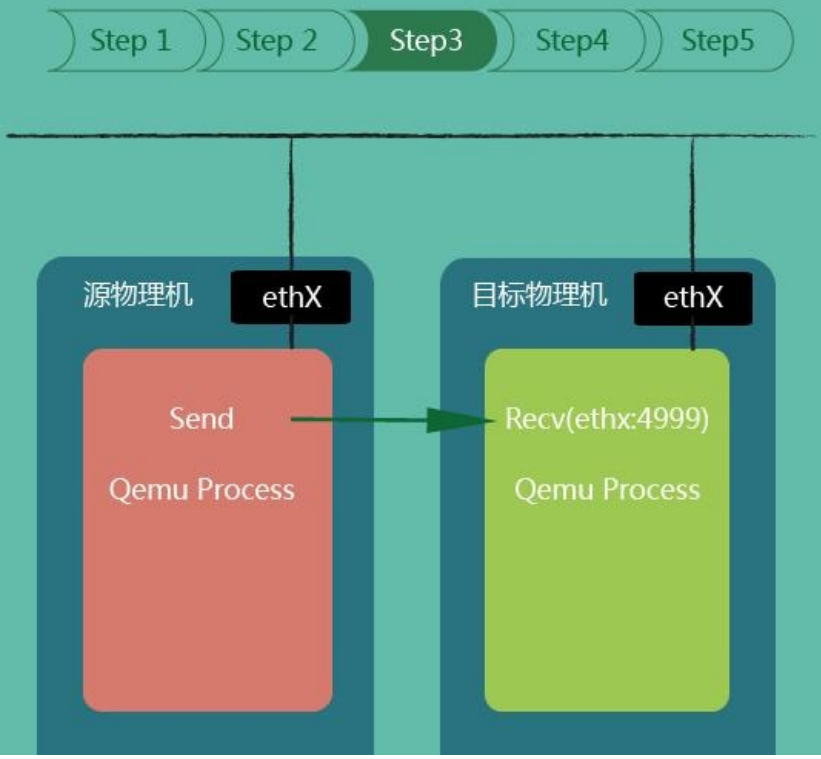


在目标物理机上创建VM-A的对应的系统盘和数据盘

在目标物理机上创建VM-A的对应的系统盘和数据盘，
两个磁盘在目标物理机上的路径和源物理机上的路径必须完全一致，
不同的是，创建的是空盘，上面没有数据。



通过虚拟化管理软件Libvirt在目标物理机上创建一个和VM-A同样配置的虚拟机VM-A' ，系统盘和数据盘使用Step.1中创建的系统盘和数据盘。VM-A' 当前状态是paused状态，虚拟cpu是暂停状态，用户态的模拟器qemu进程会监听一个内网的tcp端口，用于接收迁移的数据。



虚拟化的管理层Libvirt给VM-A对应的qemu进程发出一个迁移指令，
并指定参数，例如指定目标物理机为目标、指定需要迁移块设备、
最大的停机时间、迁移的带宽限制等等。注意，迁移数据的网络包
不是经过 vswitch, 而是直接走源物理机 ethx网卡出，
进到目标物理机的ethx，因为VM-A’ 对应的qemu进程是作为
目标物理机一个用户态进程监听在ethx对应的内网ip.
我们将Step2的图转换成qemu进程可以看得清晰一点.

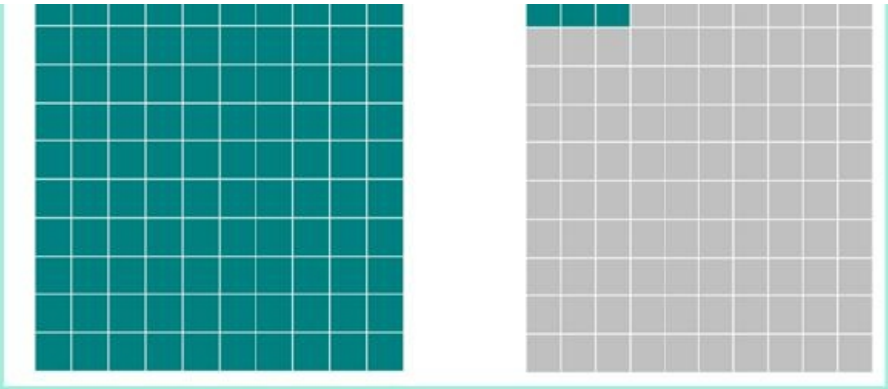
Step 1 | Step 2 | Step3 | Step4 | Step5

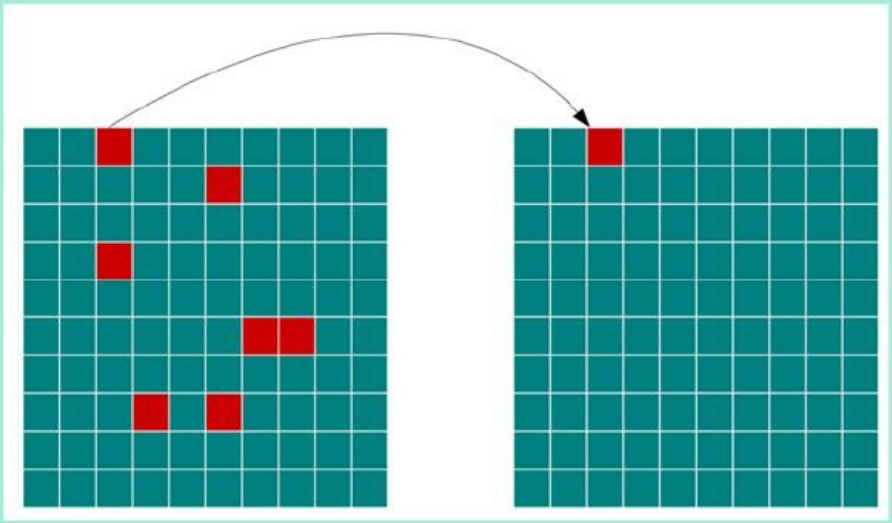
经过前面三步，虚拟机的数据就开始迁移了，剩下的挑战是怎么保证
数据的迁移是一致的，因为这个时候VM-A是运行状态，里面时刻发生
内存更新、磁盘io操作和设备状态变更，而VM-A’ 是paused是状态，
只是通过一个线程来接收VM-A进程发过来的数据，这个迁移过程
结束依据是什么呢？数据一致性和什么时候结束迁移，这两个问题
其实是一个问题，那就是虚拟机迁移完成的标准是什么？qemu里面
是这样解决的，每个虚拟机在创建的时候，都会注册块设备的在线
迁移函数、内存的迁移函数（对于设备状态而言，没有对应的在线
迁移函数，因为设备状态对应的数据量很少，只需在最后阶段一次
同步即可，qemu里面是通过每个设备注册的状态同步函数来完成
这个工作的）。

在线迁移是循环调用这些注册函数来完成源端发送数据的功能的，
注册在线迁移函数的时候，块设备比内存先注册，所以循环调用时，
会先调用块设备的迁移函数在调用内存的迁移函数，所以从现象上看，
磁盘会首先迁移，然后才迁移内存。这样依然还没有解决刚才的问题，
那就是迁移过程中发生变更的数据如何迁移?如果不迁移变更的数据，
那数据必然还是不一致的，迁移还是不能结束的。qemu的解决办法是，
每个迁移函数的实现上，分为三个阶段，第一个阶段是准备阶段，
第二阶段是迁移全量数据，第三阶段是迁移变更数据也就是脏数据。

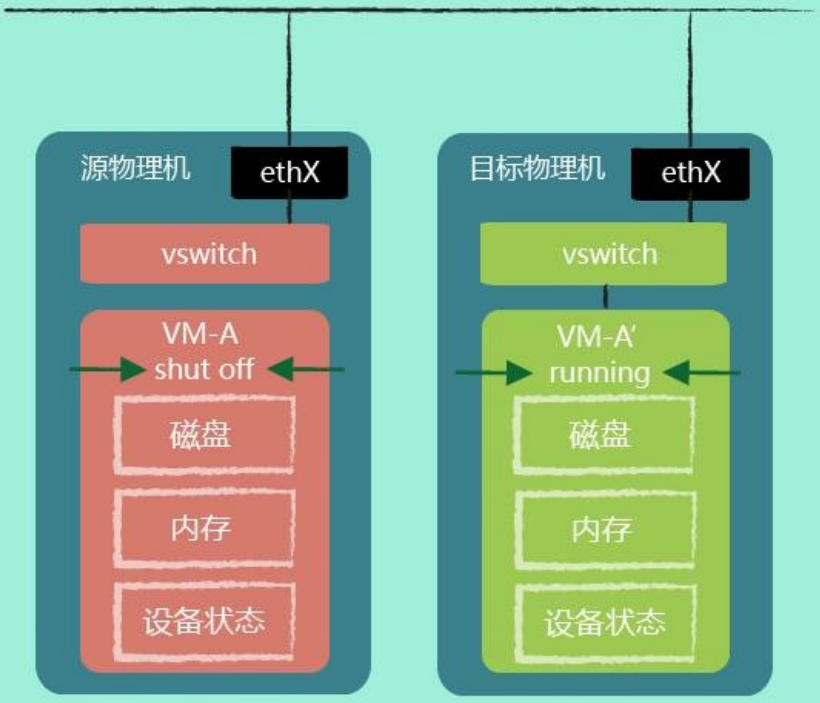
循环调用磁盘和内存迁移函数的时候，是按阶段来分别调用的。
先循环调用磁盘和内存迁移函数是第一阶段的功能，这个阶段主要是
做些传输数据的准备工作，例如把磁盘按block为单位组织成一个数组，
并设置记录脏块机制；把内存所有页全部设置为脏页，并发送开始迁移
的标志到VM-A’ 的进程。

第二阶段需要全量迁移数据, qemu在这个阶段调用磁盘和内存的迁移
函数的第二阶段功能，并且要求必须等磁盘第二阶段迁移完成后才会
执行内存第二阶段迁移，这个阶段是个迁移收敛的过程，我们前面提过，
在线迁移可以事先设置带宽和最大的停机时间，这两个参数直接影响
到收敛的过程，在第二阶段的最后，qemu边迁移边记录剩下的脏数据
大小和计算迁移过程平均传输带宽，拿脏数据/平均传输带宽的值和
最大停机时间比较，如果这个值比停机时间大，那么继续迁移，如果
比停机时间小，那么源端的qemu进程就会暂停，暂停的目的是避免
产生新的脏数据。第二阶段的过程如下面两张图所示：





暂停之后，进入第三个阶段，源端qemu进程会把磁盘和内存脏数据和设备状态一次性同步到目标端，同步完成时VM-A和VM-A’ 的数据就一致了，这个时候上层的管理软件就会把VM-A关闭，并把VM-A’的cpu恢复运行状态，整个虚拟机数据的迁移就完成了。



Step 1 | Step 2 | Step3 | Step4 | Step5

数据迁移完成后，VM-A关闭，VM-A’ 作为它的一个完全拷贝，在目标物理机上运行着，但网络还是不通的，对应Step.2的图来看，VM-A’ 通过目标物理机的vswitch 连接到物理机网卡，vswitch相当于一个虚拟的交换机，而VM-A从源物理机迁移在DestHost, 在网络上相当于把网线从一个交换机拔下插到另外一个交换机上，这个时候就需要一次arp广播，告知VM-A的mac地址已经变更到另外一台交换机的某个端口。这就是迁移完成后的网络切换，由于这个切换时间很短，少于Tcp的超时重传时间，对于原VM-A上跑着网络服务程序，几乎是无感知的。

UCloud对KVM在线迁移技术的优化

磁盘迁移的优化

UCloud的虚拟机使用的是本地存储(UDisk除外)，磁盘在物理机上是以文件方式存在的，这个文件是sparse文件，假设虚拟机的磁盘是100G, 但实际使用了10G, 那么这个磁盘文件在物理机上只占用了10G空间。但是原生的qemu在迁移磁盘时并没有保持sparse特性，所以这个100G的磁盘迁移到目标物理机上后就实际占用了100G空间，这个对物理机的磁盘空间来说是非常浪费的，因此我们给qemu打了一个patch, 让在线迁移保持磁盘文件的sparse特性，减少了对物理机磁盘的浪费，这个patch同时也减少了迁移过程传输的数据，降低带宽压力。这个patch是利用了qemu迁移磁盘时的按块迁移的机制，还有sparse文件的特点来达到这个目标的。

磁盘在迁移时是以块为单位，从源端读取一块，并发送到目标端，目标端负责写入, 而sparse文件有个特点，读取空洞块时读到的数据都是0。这样实现方案就出来了，在源端读到全0的块，就不发送到目标端，这样目标端就是跳着块来写一个文件，这样就保持了磁盘文件的sparse特性。但这还不是全部，还要考虑到原生的qemu和打了这个patch的qemu之间迁移机器如何保持sparse。解决方案也很简单，在qemu中接收磁盘数据过程判断一个块是否全部为0，如果是就不实际写磁盘，即可解决。

UDisk盘的过滤

由于UDisk是网络块存储，在迁移一台带有UDisk机器时， 迁移UDisk盘是没有必要的，因为这个盘可以通过网络挂到目标。所以我们也打了一个patch，用于迁移时过滤掉虚拟机的UDisk盘，同时对UDisk产品做了改造，支持多点挂载，这样就解决了这个问题，提高了迁移的效率。

迁移开关机确保可靠

迁移的过程是由qemu执行，更上层的控制则来自Libvirt, qemu执行完成迁移时，两边的虚拟机都是处于paused状态

qemu执行完成迁移时，两边的虚拟机都是处于paused状态。
关闭源端虚拟机，恢复目标端虚拟机其实是libvirt来控制的。
在线迁移的底线是不能因为迁移失败而导致虚拟机关机，不管成功或者失败，都要保障虚拟机实例的存活（源端或目标端）。
为了加强迁移过程的保障，避免源端和目标端关机的情况出现，我们给libvirt打了一个patch, 将libvirt中迁移过程开关机的控制逻辑移到我们自己的管理平台中。

除了上面的优化，我们还通过内核、硬件和运维对迁移进行了积极的优化。最终的效果，过去累计迁移过万次全部成功，没有发生过一次由于迁移导致虚拟机宕机的意外情况。

写在最后

未来，对于在线迁移我们还有很多工作要做，
例如减少大磁盘虚拟机在线迁移的时间，
让我们在更短的时间内能解除用户虚拟机的风险，
给我们用户提供更好的保障；
例如对于上面有内存敏感型应用的虚拟机，
如何能够在内存变更大于迁移带宽时，
能在一个有限时间内迁移完成等等。
这些都是我们努力的方向，
希望在不久的将来能给各位分享我们对这些方向的思考或者成果。