## 虚拟化的三个需求

**等价性：**

一个运行于 VMM 下的程序,其行为应与直接运行于等价物理机上的同程序的行为完全一致。

**资源控制：**

VMM 对虚拟资源进行完全控制。

**效率性：**

机器指令中经常使用的那一部分应在没有 VMM 干预下执行。

**经典的虚拟化方法：**

现代计算机体系结构一般至少有两个特权级（即用户态和核心态，x86 有四个特权级 Ring0~ Ring3）用来分隔系统软件和应用软件。那些只能在处理器的最高特权级（内核态）执行的指令称之为特权指令，一般可读写系统关键资源的指令（即敏感指令）决大多数都是特权指令（X86 存在若干敏感指令是非特权指令的情况）。**如果执行特权指令时处理器的状态不在内核态，通常会引发一个异常而交由系统软件来处理这个非法访问（陷入）**。经典的虚拟化方法就是使用“**特权解除**”和“**陷入-模拟**”的方式，即将 **Guest OS 运行在非特权级**，而将 VMM 运行于最高特权级（完全控制系统资源）。解除了 Guest OS 的特权级后，Guest OS 的大部分指令仍可以在硬件上直接运行，只有**执行到特权指令时，才会陷入到 VMM 模拟执行（陷入-模拟）**。“陷入-模拟” 的本质是保证可能影响 VMM 正确运行的指令由 VMM 模拟执行，大部分的非敏感指令还是照常运行。

**特权指令：**系统中有一些操作和管理关键系统资源的指令，这些指令**只有在最高特权级**上能够正确运行。如果在非最高特权级上运行，特权指令会引发一个异常，处理器会陷入到最高特权级，交由系统软件处理了。

**敏感指令：**操作特权资源的指令，包括修改虚拟机的运行模式或者下面物理机的状态；读写时钟、中断等寄存器；访问存储保护系统、地址重定位系统及所有的 I/O 指令。有些敏感指令可以在用户态或者内核态执行，但执行结果并不一致。

**CPU 虚拟化的一个必要条件：**

敏感指令是特权指令的子集

**解释：**x86 架构有一部分敏感指令并不是特权指令，变成了非特权指令。这就出现问题了。非特权指令在 Ring1～Ring3 上是可以执行。因为 X86 指令集中**有若干条指令是需要被 VMM 捕获的敏感指令，但是却不是特权指令（称为临界指令），因此“特权解除”并不能导致他们发生陷入模拟（也就是说该陷入 Ring0 的却仅仅陷入到了 Ring1）**，执行它们不会发生自动的“陷入”而被 VMM 捕获，从而阻碍了指令的虚拟化。VMM 不会处理这些指令。这就导致了问题。系统本想让 VMM 执行的指令却被非特权级别执行了。一个不受虚拟软件控制的指令执行在虚拟层。就好象一个学医学的和你坐在一起敲代码，你是什么感受呢？这就是问题所在了。

无法用传统方式进行虚拟化的解决办法 - **二进制翻译**

VMM 确保 Guest OS 中的的敏感指令不再执行。具体做法是代码改写，一次改写一个代码块。基本块是以转移指令结尾的一小段顺序指令序列。在执行一个代码块之前，VMM 查找其中的敏感指令并替换成 VMM 中处理例程的指令。代码块结尾的转移指令也会被替换成 VMM

管理程序的指令，以确保下一个基本块能重复此过程。翻译过的基本块基本可以缓存下来，而且大多数基本块不包括敏感操作等等，整体速度还是比较快的。