JVM

# 简介

Java 堆（每个 Java 对象在其中分配）是您在编写 Java 应用程序时使用最频繁的内存区域。JVM 设计用于将我们与主机的特性隔离，所以将内存当作堆来考虑再正常不过了。您一定遇到过 Java 堆 OutOfMemoryError ，它可能是由于对象泄漏造成的，也可能是因为堆的大小不足以存储所有数据，您也可能了解这些场景的一些调试技巧。但是随着您的 Java 应用程序处理越来越多的数据和越来越多的并发负载，您可能就会遇到无法使用常规技巧进行修复的 OutOfMemoryError。在一些场景中，即使 java 堆未满，也会抛出错误。当这类场景发生时，您需要理解 Java 运行时环境（Java Runtime Environment，JRE）内部到底发生了什么。

Java 应用程序在 Java 运行时的虚拟化环境中运行，但是运行时本身是使用 C 之类的语言编写的本机程序，它也会耗用本机资源，包括本机内存。本机内存是可用于运行时进程的内存，它与 Java 应用程序使用的 java 堆内存不同。每种虚拟化资源（包括 Java 堆和 Java 线程）都必须存储在本机内存中，虚拟机在运行时使用的数据也是如此。这意味着主机的硬件和操作系统施加在本机内存上的限制会影响到 Java 应用程序的性能。

## 本机内存简介

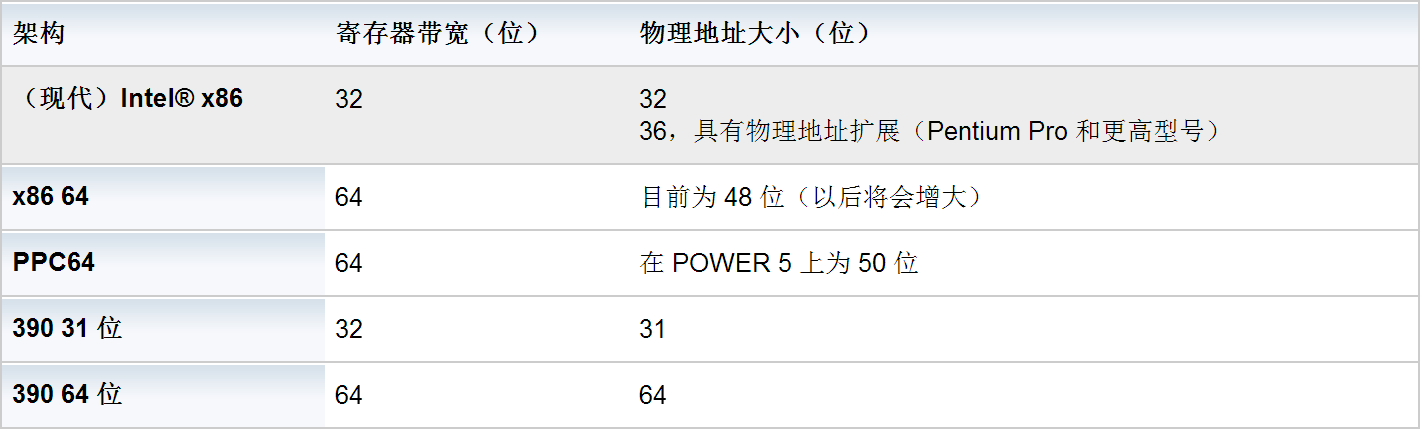
### 硬件限制

本机进程遇到的许多限制都是由硬件造成的，而与操作系统没有关系。每台计算机都有一个处理器和一些随机存取存储器（RAM），后者也称为物理内存。处理器将数据流解释为要执行的指令，它拥有一个或多个处理单元，用于执行整数和浮点运算以及更高级的计算。处理器具有许多寄存器 —— 常快速的内存元素，用作被执行的计算的工作存储，寄存器大小决定了一次计算可使用的最大数值。

处理器通过内存总线连接到物理内存。物理地址（处理器用于索引物理 RAM 的地址）的大小限制了可以寻址的内存。例如，一个 16 位物理地址可以寻址 0x0000 到 0xFFFF 的内存地址，这个地址范围包括 2^16 = 65536 个惟一的内存位置。如果每个地址引用一个存储字节，那么一个 16 位物理地址将允许处理器寻址 64KB 内存。

处理器被描述为特定数量的数据位。这通常指的是寄存器大小，但是也存在例外，比如 32 位 390 指的是物理地址大小。对于桌面和服务器平台，这个数字为 31、32 或 64；对于嵌入式设备和微处理器，这个数字可能小至 4。物理地址大小可以与寄存器带宽一样大，也可以比它大或小。如果在适当的操作系统上运行，大部分 64 位处理器可以运行 32 位程序。

下表列出了一些流行的Linux和Windows架构，以及他们的寄存器和物理地址的大小：



## 操作系统和虚拟内存

如果您编写无需操作系统，直接在处理器上运行的应用程序，您可以使用处理器可以寻址的所有内存（假设连接到了足够的物理 RAM）。但是要使用多任务和硬件抽象等特性，几乎所有人都会使用某种类型的操作系统来运行他们的程序。

在 Windows 和 Linux 等多任务操作系统中，有多个程序在使用系统资源。需要为每个程序分配物理内存区域来在其中运行。可以设计这样一个操作系统：每个程序直接使用物理内存，并且可以可靠地仅使用分配给它的内存。一些嵌入式操作系统以这种方式工作，但是这在包含多个未经过集中测试的应用程序的环境中是不切实际的，因为任何程序都可能破坏其他程序或者操作系统本身的内存。

*虚拟内存* 允许多个进程共享物理内存，而且不会破坏彼此的数据。在具有虚拟内存的操作系统（比如 Windows、Linux 和许多其他操作系统）中，每个程序都拥有自己的虚拟地址空间 —— 一个逻辑地址区域，其大小由该系统上的地址大小规定（所以，桌面和服务器平台的虚拟地址空间为 31、32 或 64 位）。进程的虚拟地址空间中的区域可被映射到物理内存、文件或任何其他可寻址存储。当数据未使用时，操作系统可以在物理内存与一个交换区域（Windows 上的*页面文件* 或者 Linux 上的*交换分区*）之间移动它，以实现对物理内存的最佳利用率。当一个程序尝试使用虚拟地址访问内存时，操作系统连同片上硬件会将该虚拟地址映射到物理位置，这个位置可以是物理 RAM、一个文件或页面文件/交换分区。如果一个内存区域被移动到交换空间，那么它将在被使用之前加载回物理内存中。图 1 展示了虚拟内存如何将进程地址空间区域映射到共享资源：