所有这类动态加载的代码，为操作系统造成了更大的复杂性，因为程序库的版本管理不是只为可执行体匹配对应版本的 DLL，而是有时把多个版本的同一个DLL加载到进程中——Microsoft称之为肩并肩（side-by-side）。单个的程序可以承载两个不同的DLL，每个可能要加载同一个Windows库——但对该库的版本有不同要求。

较好的解决方案是把代码放到独立的进程里。而在进程外承载的代码结果具有较低的性能，并在很多情况下会带来一个更复杂的编程模型。微软尚未提供在用户态下来处理这种复杂度的一个好的解决办法。但这让人对相对简单的内核态产生了希望。

该内核态具有较少的复杂性，是因为它相对于用户态提供了更少的对外部设备驱动模型的支持。在 Windows 中，系统功能的扩展是通过编写用户态服务来实现的。这对于子系统运行得很好，并且在只有很少更新的时候，而不是整个系统的个性化的情况下，能够取得更好的性能。在内核实现的服务和在用户态进程实现的服务之间只有很少的功能性差异。内核和过程都提供了专用地址空间，可以保护数据结构和服务请求可以被审议。

但是，可能会与服务的用户态处理内核中服务有重大的性能差异。通过现代的硬件从用户态进入内核是很慢的，但是也比不上要来回切换两次的更慢，因为还需要从内存切换出来进入另一个进程。而且跨进程通信带宽较低。

内核态代码（非常仔细地）可以把用户态处理的数据作为参数传递给其系统调用的方式来访问数据。通过用户态的服务，数据必须被复制到服务进程或由映射内存等提供的一些机制（Windows Vista中的ALPC功能在后台处理）。

将来跨地址空间的切换代价很可能会越来越小，保护模式将会减少，或甚至成为不相关。在Singularity中，微软研究院（Fandrich等人，2006年）使用运行时技术，类似 C# 和 Java，用来做一个完全软件问题的保护。这就要求地址空间的切换或保护模式下没有硬件的切换代价。

Windows Vista利用用户态的服务进程极大地提升了系统的性能。其中一些服务是同内核的组件紧密相关的，例如 lsass.exe这个本地安全身份验证服务，它管理了表示用户身份的令牌（token）对象，以及文件系统用来加密的密钥。用户态的即插即用管理器负责确定要使用新的硬件设备所需要的正确的驱动程序来安装它，并告诉内核加载它。系统的很多功能是由第三方提供的，如防病毒程序和数字版权管理，这些功能都是作为内核态驱动程序和用户态服务的组合方式实现的。

在Windows Vista中 taskmgr.exe 有一个选项卡，标识在系统上运行的服务。（早期版本的 Windows 将显示服务使用 net start 命令的列表）。很多服务是运行在同一进程（svchost.exe）中的。Windows 也利用这种方式来处理自己启动时间的服务，以减少启动系统所需的时间。服务可以合并到相同的进程，只要它们能安全地使用相同的安全凭据。

在每个共享的服务进程内，个体服务是以DLL的形式加载的。它们通常利用Win32的线程池的功能来共享一个进程池，这样对于所有的服务，只需要运行最小数目的线程。

服务是系统中常见的安全漏洞的来源，因为它们是经常是可以远程访问的（取决于TCP/IP 防火墙和 IP 安全设置），且不是所有程序员都是足够仔细的，他们很可能没有验证通过 RPC传递的参数和缓冲区。

一直在Windows中运行的服务的数目是令人惊讶的。但这些服务中的很少一部分不断收到单个请求，如果有，那这样的进程看起来就像是远程的攻击者试图找到系统的漏洞。结果是越来越多的服务在Windows中被默认为是关闭的，特别是Windows Server的相关版本。

11.4 Windows Vista中的进程和线程

Windows 具有大量的管理CPU和资源分组的概念。以下各节中，我们将检查这些有关的 Win32 API 调用的讨论，并介绍它们是如何实现的。

11.4.1 基本概念

在 Windows Vista 中的进程是程序的容器。它们持有的虚拟地址空间，以及指向内核态的对象的线程的句柄。作为线程的容器，它们提供线程执行所需要的公共资源，例如配额结构的指针、共享的令牌对象以及用来初始化线程的默认参数——包括优先次序和调度类。每个进程都有用户态系统数据，称为