64位windows常规编程简介 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/140/2021_2022_64_E4_BD_8Dwindo_c100_140190.htm 性能和可伸缩性 为了可以伸缩，您必须了解这对于您特定的方案意味着什么。例如，对于 Web 服务器，可伸缩性意味着可以为与所连接的用户数量相关的页面提供服务。请将其考虑为线图表。 图1.线性比例示例 随着用户数量的增加，每秒的页面数量也应该增加。上图显示了一个线性比例。当用户数量达到 3 倍时，每秒提供的页面也相应增加。 另一个比例的定义与硬件相关。如果我将系统上的处理器数量增加一倍，那么我的 Web 服务器是否也会将输出增加一倍呢？RAM 或磁盘等情况如何呢？应用程序也需要根据这个想法设计。您所创建的线程数量应该基于系统中的处理器数量以及每个线程进行的工作类型。用于缓存网页内容的内存量应该与可用于应用程序的内存量成一定比例，等等。这个概念通常被称为“ 向上扩展 ”。如果我将框构建得越来越大，那么可以相应的生成越来越多吗？ 伸缩的其他形式是当您谈论分布式计算或服务器场时。这通常被称为“ 向外伸缩 ”。如果我将服务器场中的计算机数量增加一倍，那么我的输出也会增加一倍吗？ 当设计可伸缩的系统时，需要考虑这些方案。当前，硬件变得越来越大（Itanium 最多支持 64 个处理器计算机），因此向上扩展需要在开发人员头脑中处于最重要的位置。如果您的图表转为水平、甚至随着您增加资源开始下降，这尤为正确。如果整个系统的某个部分无法伸缩，它可能会对整个系统产生负面的影响。 线程：如何有效地使用它们 在线程间分割您的工作可以简化代码，在

多处理器系统上可以使您的代码更有效，但是如果您不知道自己在做什么的话，还会降低性能和可伸缩性。例如，如果应用程序中的所有线程都需要获得相同的全局关键部分，那么对该关键部分的争用可能会使您的线程花费其大部分休眠时间。它还可能导致发生过多的上下文转换，进而可能会引起应用程序占用系统内核中相当比例的处理时间，甚至根本没有运行您的代码。如果在多处理器的系统上，这些问题会尤其糟糕，您额外的处理器可能会结束当前闲置，等待访问共享数据。 要使用的线程的理想数量等于系统中处理器的数量。如果您的线程相互独立并受到处理器的限制，那么它们应该能够每次都消耗掉其整个时间片。如果您具有可能执行阻止操作的线程，那么您可能希望增加线程的数量，以便当一个线程休眠时，另一个线程可以取代其位置。您将要确定线程阻塞的位置及频率。意识到这一点后，您就可以知道应该运行的线程数量。您始终要为每个处理器准备好一个线程。否则，您就浪费了处理能力。当然，这些仅仅是指导原则，并且确定应用程序是否以尽可能高的效率运行的唯一方法就是对其进行分析和测试。 异步 I/O：不会阻塞等待数据 基于 NT 内核的 Windows 系统支持异步 I/O，又称重叠的 I/O.大多数形式的 I/O 都可以异步完成。这包括文件 I/O 和网络 I/O.对于文件 I/O，您可以使用 ReadFile/WriteFile API.当读/写时，通过借助 FILE_FLAG_OVERLAPPED 标记打开文件并指定 OVERLAPPED 结构，您将使系统在 I/O 完成时通知您。这使您可以在等待的过程中完成其他工作。对于使用 Windows Socket （WinSock） 的网络 I/O，您可以使用 WSASocket 创建套接字，并指定 WSA_FLAG_OVERLAPPED 标记，然后当调

用 WSARecv/WSASend API 时，您可以指定一个 OVERLAPPED 结构或一个回调函数。当您编写网络服务器时，异步 I/O 尤为有效。您可以将多个接收请求“ 排入队列 ”，然后去休息，等待其中一个完成。当一个完成时，就会处理传入的数据，然后将另一个接收“ 排入队列 ”。这比使用 0select API 来轮询数据好得多，并且它可以更有效地使用系统资源。 等待异步 I/O 请求完成有几种选项： 调用 GetOverlappedResult API 在发出异步 I/O 请求后，您可以使用 GetOverlappedResult API 来轮询请求的状态，或者仅仅等待请求的完成。当请求完成时，GetOverlappedResult 将返回请求过程所传输的字节数。 使用 HasOverlappedIoCompleted 宏 您可以使用 HasOverlappedIoCompleted 宏来有效地进行轮询与 OVERLAPPED 结构相关联的请求是否已经完成。请求完成后，您就可以使用 GetOverlappedResult API 来获得有关请求的更多信息（例如传输的字节数）。 指定 OVERLAPPED 结构中的事件 通过在 OVERLAPPED 结构的 hEvent 字段中指定一个事件，您可以执行自己的轮询或等待请求的完成，方法是在对 WaitForSingleObject 或 WaitForMultipleObjects 的调用中指定一个事件。当重叠操作完成时，内核将发信号通知该事件。 将内核对象绑定到 I/O完成端口 I/O 完成端口是系统提供的非常有用的工具。有关信息，请参阅下面的部分。对于事件驱动的系统（例如网络服务器等待输入），I/O 完成端口提供了用于等待和处理传入事件的完美机制。 