工作者工厂(线程池)

工作者工厂(worker factory)是一种用于实现用户模式线程池的内部机制。老版本的线程池例程完全是通过Ntdll.dll库在用户模式下实现的。此外Windows API还为开发者提供了多个可调用的函数，这些函数提供了可等待计时器(CreateTimerQueue、CreateTimerQueueTimer和类似计时器)、等待回调(RegisterWaitForSingleObject)，并根据所处理的工作量提供了与自动线程创建和删除(QueueUserWorkItem)有关的工作项处理能力。

老版本的实现会面临一个问题：一个进程中只能创建一个线程池

Windows中这种内核线程池功能由名为TpWorkerFactory 的对象管理器类型管理，此外参与工厂和工作者管理的还包括4个原生系统调用(NtCreateWorkerFactory、NtWorkerFactoryWorkerReady、NtReleaseWorkerFactoryWorker 和 NtShutdownWorkerFactory)、两个查询/设置原生调用(NtQueryInformationWorkerFactory和NtSetInformationWorkerFactory),以及一个等待调用(NtWaitForWorkViaWorkerFactory)。与其他原生系统调用类似，这些调用为用户模式提供了指向TpWorkerFactory对象的句柄，其中包含了诸如名称和对象属性、所需访问掩码以及安全描述符等信息。然而与其他系统调用被 Windows API包装的做法不同，线程池的管理是由 Ntdll.dll的原生代码负责的。这意味着开发者所面对的是一种不透明的描述符:指向线程池的TP\_POOL指针，以及指向池中所创建对象的其他不透明指针，包括TP\_WORK(工作回调)、TP\_TIMER(计时器回调)、TP\_WAIT(等待回调)等。这些结构保存了各种信息，例如指向TpWorkerFactory对象的句柄。

工作者工厂的实现需要负责分配工作者线程(并调用特定用户模式工作者线程的入口点)，维护最小和最大线程数(以便实现永久工作者池或完全动态的池)，并负责其他记账信息。借此，诸如关闭线程池等操作即可通过到内核的一次调用实现，因为内核已经成为负责线程创建和终止的唯一组件。

由于内核会按需动态创建新线程(根据所提供的最小和最大线程数)，这也可以提高支持全新线程池实现的应用程序的可伸缩性。当下列所有条件全部满足时，工作者工厂就会新建线程。

1. 动态线程创建已启用。
2. 可用工作者数量低于工厂所配置的工作者数量最大值(默认为 500 个)。
3. 工作者工厂包含绑定对象(例如工作者线程正在等待的ALPC端口)，或线程已经被激活入池。
4. 相关联的工作者线程存在等待处理的I/O请求包(I/O Request Packet，IRP)

当线程变为空闲状态(即已经不再处理任何工作项)并持续超过10s(默认值)时线程还会被终止。此外，虽然在老版本的实现中，开发者始终可以尽可能使用更多的线程(取决于系统中的处理器数量)，但现在应用程序可以通过线程池自动从运行中新添加的处理器获益。这是通过Windows Server 对动态处理器的支持实现的。

工作者工厂的创建

工作者工厂实际上只是一种“包装”，借此管理原本需要在用户模式下(以牺牲性能为代价)处理的各种任务。从架构来看，新线程池代码的大部分逻辑依然位于 Ntdll.dll中。此外，可伸缩性、内部等待机理以及更高效的工作处理，这些并非工作者工厂的代码提供的，而是一些古老的 Windows 组件 [I/O 完成端口，或更确切来说是内核队列(KQUEUE)] 共同的功劳。实际上，在创建工作者工厂时，必须首先由用户模式创建一个I/O完成端口并传入其句柄。

I/O 完成端口（IOCP）是一种高效的任务调度和通知机制，用于处理异步 I/O 事件。IOCP 可以在高并发场景下实现高性能，因为它将线程与 I/O 操作分离，并在操作完成时通知线程。这种机制允许线程在等待 I/O 完成时执行其他任务，从而提高了整体性能。

内核队列（KQUEUE）是另一种用于处理和调度事件的机制。KQUEUE 提供了一种将多个事件（如文件描述符和进程事件）组合在一起并等待它们发生的方法。KQUEUE 可以用于实现高效的事件驱动编程，因为它使应用程序能根据事件的发生而执行相应的操作。

正是借助这个I/O 完成端口，用户模式的实现才得以将工作放入队列并等待，只不过此时需要调用工作者工厂系统调用，而非 IO 完成端口API。然而从内部来看，“释放”工作者工厂调用(用于将工作放入队列)是一种围绕 IoSetIoCompletionEx 进行的包装，借此增加等待处理的工作数量;同时“等待”调用也是一种围绕 IoRemoveloCompletion进行的包装。

工作者工厂由执行体函数 NtCreateWorkerFactory 创建而来，可接受多种参数进而对线程池进行定制，例如要创建的线程数最大值、初始已提交大小以及保留的栈大小。然而CreateThreadpool 这个 Windows API 会使用可执行映像内的默认大小(与默认CreateThread的做法类似)。不过Windows API无法提供覆盖这些默认值的方式。

NtQueryInformationWorkerFactory API可以转储工作者工厂结构中几乎所有的字段。