借助 LabVIEW 应对多核编程的挑战

本文列举了多核编程开发人员们所面临的挑战，并且着重介绍了 NI 公司 LabVIEW 图形化编程环境的特点，它使工程师与科学家们能够以更简便的方式进行多核编程。具体说来，本文主要介绍了设计并行化应用程序构架、处理线程同步和调试多核程序等内容。

多核技术的未来

在过去几年，处理器制造商们一直通过单纯地提高时钟频率来提高 CPU 性能。然而近年来，处理器技术的最新趋势变为了更多的核。也就是说处理器制造商们正在把若干个 CPU 封装在一块芯片上。除了目前已经相对成熟的双核与四核处理器，Intel 公司甚至已宣称将在五年内推出 80 核的处理器！

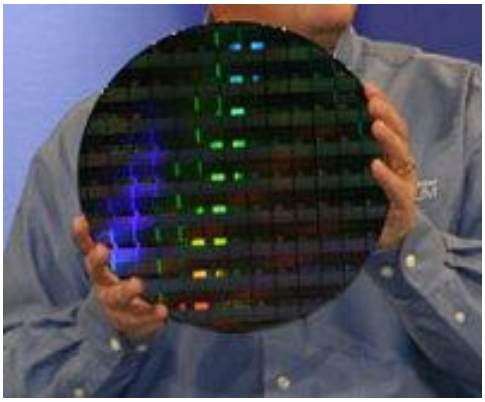

图 1 – CEO Paul Otellini 展示了英特尔公司的 80 核处理器原型(slashgear.com 提供)

似乎这种运算性能的无限提升让我们可以运行最复杂的应用程序——这听起来真是令人兴奋。同时，利用这种性能的巨大提升，您可以想像一下现有的程序将运行得多快！你也许会怀疑这种事情并不会这么简单吧？那么你猜对了，事实上确实如此。

在多核处理器上开发程序要比在单核处理器上要困难得多。虽然多个应用程序可以方便地运行在不同处理器上，但是程序员们必须非常谨慎地编写程序代码，以有效利用多核技术。这意味着，在现今的多核处理器上运行现有应用程序，性能几乎不会得到提升，即使有性能提升也是很小的。

开发并行化代码

编写并行程序的首要挑战就是，确定程序中哪些部分能够并行地执行，然后通过代码予以实现。我们将这些可以并行执行的代码称为线程。因此，整个并行程序也被称为多线程应用程序。

对于传统的基于文本编程的程序员来讲，他们需要在应用程序中使用诸如 OpenMP 或 POSIX 等 API，来明确定义这些线程。这是因为基于文本的编程方式原本是串行执行的，所以要对并行的多线程代码予以形象的表述是十分困难的。另一方面，利用 NI LabVIEW 的图形化特点，程序员们可以很容易地编写和表述并行程序代码。

不仅如此，LabVIEW 还可以为代码中的并行部分自动生成线程，这个优点可以很好地帮助相对缺少或者没有编程背景的工程师和科学家们将更多时间用在解决实际问题中，而无需关注应用程序的底层实现。

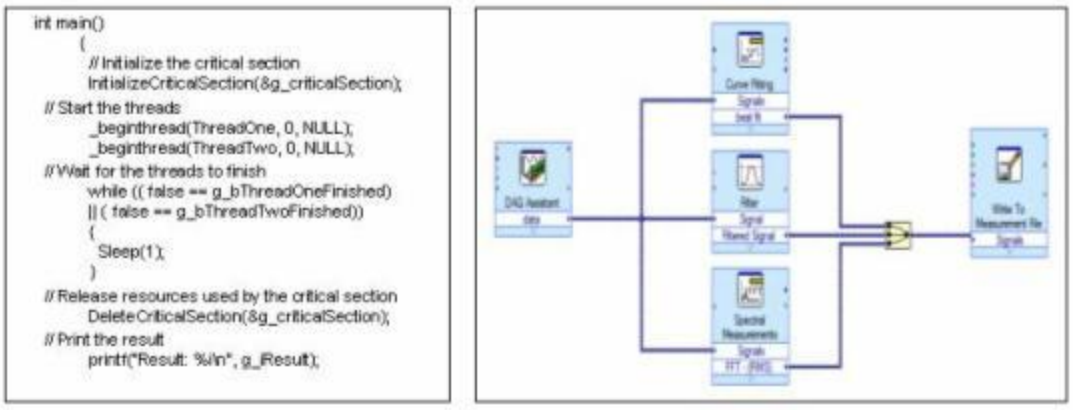

图 2 — LabVIEW 和基于文本语言的多线程编程对比

线程同步

其次，多核编程的第二个挑战就是线程同步。当某个应用程序中有成千上百个线程在运行时，我们必须确保所有线程都能够协调地工作。例如 ，如果两个或两个以上的线程在同一时间企图访问同一个内存地址，那么将会发生数据冲突。显然，在应用程序中识别可能产生冲突的代码是一项艰巨的任务。

但是，通过在 LabVIEW 中以图形化的方式创建程序框图，您可以迅速开始实现具体的任务，将构想变为现实，而无需考虑线程同步的问题。图 2 给出的 LabVIEW 程序中，两个并行的图形化代码段在写入文件时都需要访问硬盘，而 LabVIEW 可以自动地处理这两个线程的同步。

图 3 — LabVIEW 中的自动线程同步功能的简单展示

调试

大多数的程序在第一次运行时都无法按照开发者所想地来运行。这种情况无论对于基于单核还是多核编写的应用程序来说都是一样。为了在逻辑上确定代码段中的功能性错误，用户必须在开发环境中利用有效的调试工具来确保应用程序执行正确。

对于多核程序来说，调试必然带来了更大的挑战：您不仅需要同时跟踪多段代码的运行，还需要确定每段代码在哪个核上运行。除此以外，如果您经常编写多线程应用程序，还需要处理线程交换和线程饥饿等问题，这些都是需要在调试阶段确定的。

LabVIEW 中则包含了若干可以极大简化多核程序调试复杂性的特性。具体而言，您可以利用高亮化执行工具，

将程序并行化执行的过程快速而简单地以图形化的方式进行显示(这是由于 LabVIEW 本质上是基于数据流的)。例如图 3 中的简单应用程序，在打开高亮化执行功能后，您可以容易地将并行代码的执行过程以图形化的方式显示出来。

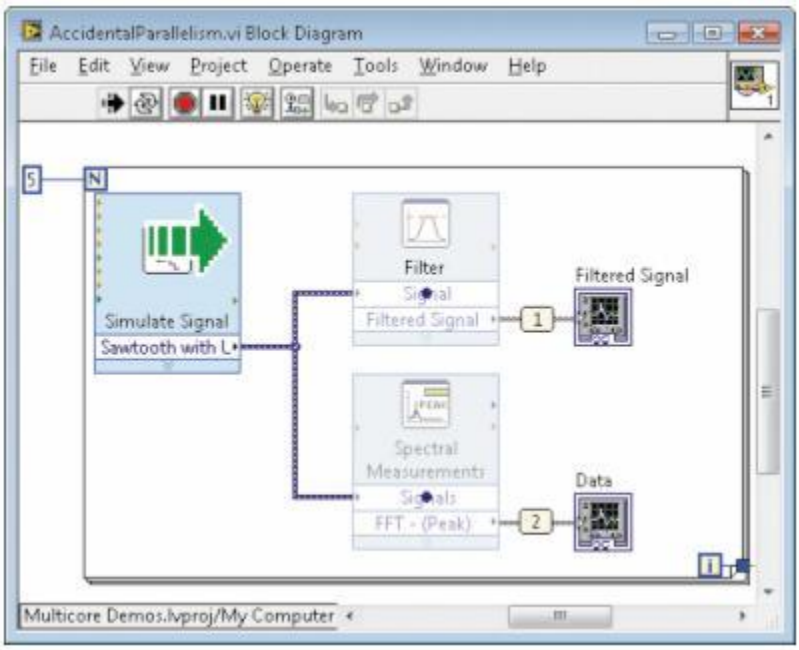

图 4 — 在 LabVIEW 开发环境中对代码运行进行图形化高亮执行

此外，LabVIEW 实时模块同时还提供了在多核处理器上进行确定性执行的能力以及更细致的调试功能。作为 LabVIEW 的新成员，Real-Time Execution Trace Toolkit 允许程序员们以图形化的方式显示指定线程是在哪个核上运行的，从而迅速确定线程饥饿和交换等问题。

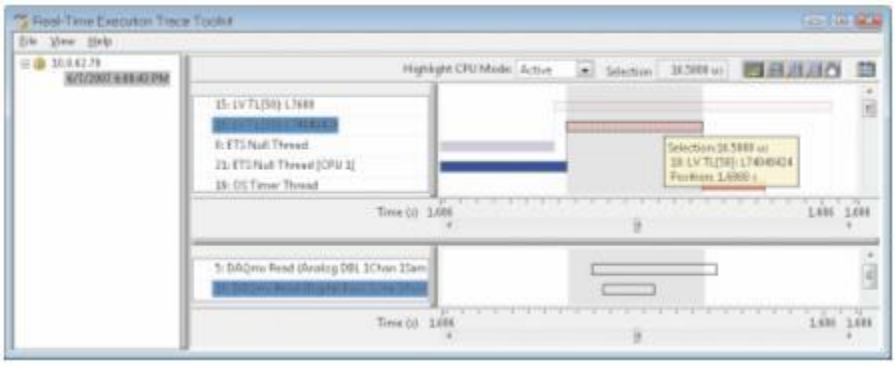

图 5 LabVIEW 8.5 执行跟踪工具包

总结

总而言之，熟悉多核处理器编程的程序员们需要考虑多线程所特有的挑战，包括(并行化应用程序的构架、线程同步与调试等等。相信随着芯片上处理器核数的不断增加，在多线程应用程序中使用正确的并行编程技术将变得越来越重要。