利用动态时间分区发挥 CPU 最佳效率

QNX 软件系统公司产品经理郑怡

简介

虽然高功效的多核处理器已经面世，系统设计人员仍然必须确保各个进程正确共享资源。*时间分区*是目前可确保一个操作系统能在多个竞争进程中正确共享资源的关键技术，它甚至可以保护计算机资源免受恶意软件侵害。

时间分区可以为*静态*或者*动态*。某些操作系统还支持*空间分区*，保证每一个分区拥有一定的内存量。但是，空间分区将不在本文讨论之列。

关于时间分区

*时间分区*将既定的 CPU 时间（即周期）分配为逻辑区间，也即分区。分区由一个或多个线程组成，这些线程不一定来自同个进程。实际上，多线程的进程可将其线程分配给不同的分区。每个分区可获得预先规定的 CPU 时间，并且分配到的量不需要相同；任何逻辑分区在没有 100%消耗 CPU 时间的情况下，都可以正常工作。

当给一个分区分配到 CPU 时间之后，就产生了该分区的*哪个*线程需要调度这个问题。尽管任何调度算法都可以采用，比如轮叫调度，但最好还是采用基于优先级的调度算法。分区及优先级调度的分离既能确保每个主要任务区域（即分区）在周期内都分配到一定量的 CPU 时间，又能按照每个线程在系统内的整体重要性来分配分区内线程的实际工作。

举例来说，一个分区可以包括提供用户界面的进程中的线程，另一个分区可以包括控制多媒体（以确保提供连续的音乐或者视频播放）的进程中的线程，还有一个分区可以及控制其他数字处理功能的进程，以实现满意的用户体验。如果这是工厂自动化系统，则分区可以是用户界面、电机控制、传感器处理及与中央控制计算机的通信。限制该组合的仅仅是操作系统允许的分区数量、整体系统要求以及系统架构师的想象力。

无论选择何种设计，将系统的不同分部置于不同的分区可以防止系统的某一部分独占计算机资源，造成其他部分资源匮乏。

静态时间分区

在静态时间分区的调度机制中，每个分区预先分配到一个周期内的定量 CPU 时间，这些分配不可在运行时间改变。并且，每个分区的线程分配在系统设计时便已完成，不能改变。这通常可见于航空电子系统，ARINC653 标准也指定这种机制的使用。

但是，静态时间分区有其劣势，这包括：

- *难以决定适当的分区大小及为这些分区分配线程。*程序员如果不使用诸如 RMA 之类的形式化分析方法，则必须依赖“直觉”进行实验调试。在运行时尽管可以观察到系统操作，也不能进行任何更改。
- *未使用的时间即为浪费的时间。*如果某个分区用不完其名下的 CPU 时间，调度程序也无法将这些时间分配给其他分区。直到下一个分区被调度之前，此分区剩余的时间段中，，CPU 都为闲置状态。

动态时间分区

使用动态时间分区，操作系统可以更改各分区在处理周期获得的 CPU 时间，以实现 CPU 利用率的最大化。如果某个分区的线程用不完所有分配到的 CPU，则剩余的时间可以分配给其他任务。动态时间分区有时被称为“偷闲”，这种说法不太精确，因为“偷闲”调度仅仅是动态时间分区调度方法的一种，还有其他类型存在。

本文介绍的是我们最为熟悉的*自适应分区*技术，由 QNX Neutrino 实时操作系统实现的动态分区技术。自适应分区技术将分区闲置的 CPU 时间重新分配至其他分区，最大程度优化资源有限型设备的性能。自适应分区技术尤其适用于突发性 CPU 需求。如果系统处于轻载状态，则调度原理与优先级驱动的系统类似；如果系统处于重载状态，则调度原理与静态时间分区类似。并且，与很多系统的时间分区不同的是，自适应分区技术允许程序员重新将线程分配给其他分区，并动态调整分区规模（单个周期里，CPU 分配给分区的规模）。

有些安全关键型的系统要求使用静态时间分区。自适应分区系统可以轻易地转换成静态时间分区系统。在每个分区内放置一个包含简单无限循环的最低优先级线程的，便可以迫使分区用*完*所有分配的时间，而将系统转变为静态时间分区系统。但是，静态时间分区的系统不能转变为自适应时间分区系统。

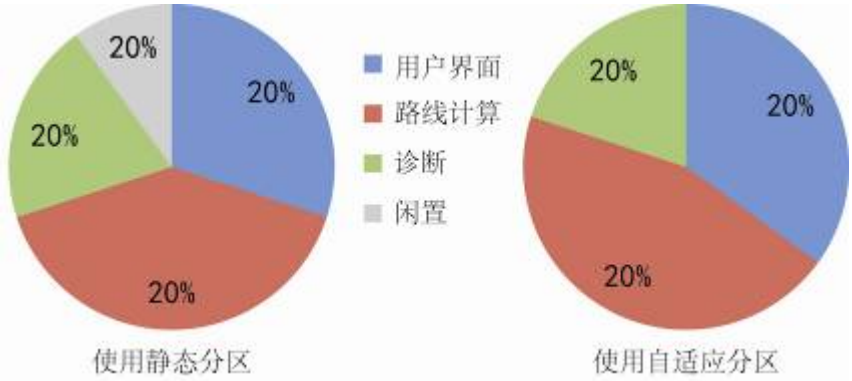
示例 1：全球定位系统

让我们来比较一下 GPS 如何在静态和自适应时间分区下工作。假设共有三个分区，下图将以不同的颜色代表，以识别这些分区和闲置时间。

名称	时间分配
用户界面	30%
路线计算	40%
诊断和数据获取	30%
闲置	N/A

重布路线

重布路线时，GPS 需要决定新的路线，并进行展示，因此 前两个分区可能会超出其分配的时间。使用静态分区，第三个分区用不完的时间都会成为闲置时间，如图 1a 所示。



使用自适应分区，剩余的时间会分配给其他分区，使系统响应更快，更加有效地利用资源，如图 1b 所示。

启动

当给 GPS 启动时，自适应分区的作用更加显著。在这种情况下，路线计算分区为闲置，而用户界面分区也较为清闲。在静态分区下，CPU 使用可能如图 2a 所示。

使用自适应分区，其他分区不用的时间将分配给诊断和数字获取这一分区，如图 2b 所示。

示例 2：简单的自动化系统

在另一个例子里，让我们观察一下带有多个元件的简单自动化系统：

元件	线程优先级
远程网络监控模块	低
本地人机界面（HMI）	中低

传感器扫描和数据获取	中低
电机控制	高

如果我们只依靠该系统里的线程优先级，则在集成阶段，我们可能会发现在操作员使用本地 HMI 前，远程监控模块都会正常工作。一旦操作员开始使用本地 HMI，远程模块就会冻结，并停止显示更新。追寻故障原因可能会发现，当 HMI 命令导致大量电机控制操作时，远程模块无法再获取 CPU 时间。

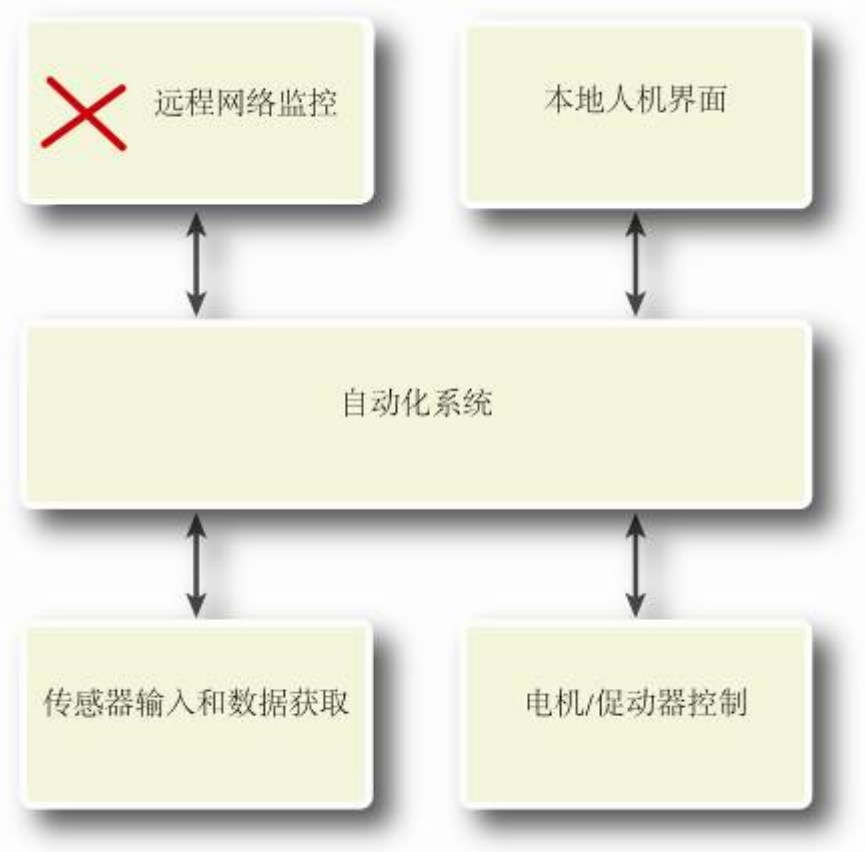


图 3：在一个基于优先级的系统中，当本地 HMI 繁忙时，远程网络控制得不到 CPU 时间。

通过调整优先级来解决问题

为了修复系统，我们可能为本地 HMI 指定一个低于远程监控模块的优先级，但这可能会导致 HMI 性能无法达标。将远程模块、数据获取和 HMI 的优先级都设定为中等也不能解决问题，因为这会影响数据获取的性能。既然重新设定优先级解决不了问题，我们必须改变线程运行状态，监控 HMI 的 CPU 消耗，并确保它间或让远程监控运行——这会大大增加集成阶段的成本。

利用分区进行设计

时间分区提供更好的解决方案。我们将 CPU 预算分配给子系统和每一个设计团队，无需全系统范围的优先级机制。每一个团队可以在自己的分区内开发自己的优

先级方案。实时操作系统保证分区的预算，并在每个分区内使用基于优先级的调度程序。按照需要的不同，这些分区可以是静态或者自适应。

分区	时间分配
传感器输入和数据获取	30%
本地 HMI	10%
远程网络控制	10%
电机/促动器控制	50%

时间分区是否永远是最佳方案？

有时候，时间分区也会存在问题；比如说，有人按了电源开关。在这种情况下：

- 循环调度程序需要时间来意识到事件的发生
- 实时期限仍然需要满足
- 使用中断进程来处理事件可能会出问题

这种情况对静态和自适应分区调度都有影响，但是自适应分区调度所受的影响可能较小。就拿 QNXNeutrino 实时操作系统来说，有一个*关键线程*的概念存在；当这个线程被激活时（比如当有人按了紧急电源切断按钮），调度程序会停止所有分区调度，并立即着手处理该线程。

关于QNX 软件系统公司

QNX 软件系统公司是 Research In Motion 公司（RIM）的子公司，是嵌入式系统市场上操作系统、中间件、开发工具和专业服务的领军者。包括思科、戴姆勒、通用电气、洛克希德•马丁和西门子在内的众多全球知名技术领先企业，都将 QNX 技术应用在网络路由器、车载远程信息处理装置、工业控制系统、医疗设备、安全防卫系统和其他任务关键性和生命关键型应用中。QNX 软件系统公司成立于 1980 年，总部位于加拿大渥太华，其产品行销全球 100 多个国家或地区。