WIND RIVER

多核挑战与机遇：选择最佳解决方案

Tomas Evensen
风河公司首席技术官

提要

许多技术和市场趋势正在彻底改变未来的设备架构和开发方式，主要包括：芯片运行频率已经接近物理极限；将多核技术用于硬件负载分流（Offload）；将多处理器系统集中化为多核架构；在嵌入式设备中运用Hypervisor；使用“封装”来达到特定的性能目标、在特定的设备融入更多的功能；又或者，为了追求更高的安全性和保密性，需要采用经过认证的系统。

多核和虚拟化有望使上述趋势化为现实。本文会更加明了地呈现这些趋势。

商业和技术趋势

频率极限

留意一下芯片厂商处理器产品路线图的右上角，最常看到的就是多核处理器。多核意味着在芯片内拥有一个以上的处理器内核或CPU。为什么几乎所有的芯片厂商突然都同时转向了多核？

这一切都归结于物理上的原因。在很长一段时间内，芯片厂商可以通过提升时钟频率来稳步实现越来越快速的处理器。这种方式不可避免地会增加功耗，但与此同时，另一个发展趋势是芯片线宽越来越小，这反过来要求降低功耗。这两种趋势在几年前开始出现严重的冲突。不过，人们对更高性能的处理器的需求一点都没有停歇，于是处理器架构设计师不惜投入更多的电路、运用很多技巧来追求性能提升，例如超流水（super-pipelining，每条指令通过更多环节以便更快地执行）、超递增（super-scaling，每个周期内送出多条指令）、分支预测（branch prediction，通过预测跳转路径将超长流水操作的影响降到最低）、寄存器重命名（register renaming，在即时流水步骤期间使用虚拟寄存器，实现更多的并行操作）等等。

现在，我们已经面临着这样的困境——提升处理器频率或者提高连续代码流的并行处理能力将会耗费大量成本，但却收效甚微。此外，即使你能够获得少量的性能提升，处理器功耗却急剧增长。

导致这种现象出现的真正因素有三方面：第一，功耗增长与频率提高程度直接成正比；第二，需要更高的电压来实现频率的提升，而功耗增长与电压增长的二次方成正比；第三，处理器时钟频率越快，CPU速率和总线速率之间的差距就越大，就需要通过采用前述的手段和更大的缓存，从而大大增加了晶体管数目，而这些大量增加的晶体管最终导致更高的功耗。

图1展示了当频率增大时功耗的变化情况。第一对柱状图对比显示了在处理器在全速率运行下的性能和功耗；第二对柱状图对比显示了当CPU时钟频率降低到最高值的80%，功耗虽然降低了几乎一半，却还能保持大部分的性能；最后一对柱状图对比显示了如果你加入了第二

个内核并且同样保持在80%频率运行，响应的功耗与单个CPU最高时钟速率下基本相同，但性能却提升了约60%。这至少是理论结果。

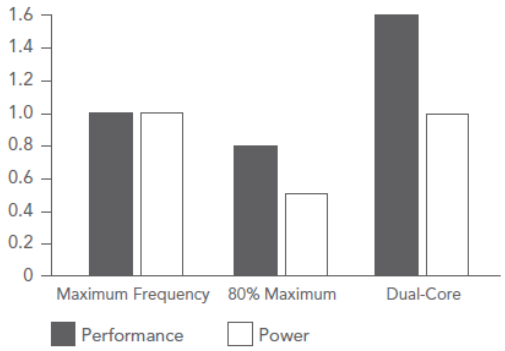

图1：不同频率下性能与功耗对应关系

这听起来很棒！为什么我们之前没有这么做？实际上，我们以前就做过，多进程技术的运用可以追述到数十年前。所不同的是，当时采用的是多芯片甚至多板卡，而主要目的是为了追求更高的性能。直到最近，把多个内核置入同一芯片的技术和原因才逐渐出现。

从硬件角度来看，这是一个绝佳的解决方案——更高的MIPS、更低的功耗。问题在于我们很难利用这些提升的性能。在过去的每一代处理器技术中，都无需对程序进行重大改变，它们自然就能运行的越来越快。在加入更多的处理器内核来获取更高性能的同时，也需要以多线程并行运行来利用这些性能。对于绝大多数嵌入式应用，处理器在多数时候都是运行一个线程。当在系统中加入了一个新的处理器，原有线程仍然让第一个CPU繁忙不堪，而新加入的处理器却处于空闲状态。

从多核架构中获取最大性能的困难在于，应用中的每个任务都必须并发地执行起来。通常情况下，找出并行性是一个十分困难的问题，但在一些特殊的情况下，这个问题却很容易解决。

采用多核技术进行硬件Offload

采用多核技术的一个成功之处就是可以执行原本由专用处理器、FPGA和ASIC等专用硬件来处理的特定任务。由于这些任务原本就是由独立的硬件单元执行，并行化的问题原本就不存在。针对特定任务，与旧的传统方法相比，使用较低档的硬件支持多个较慢速的内核，可以获得更好的并行性能，并且更易于设计和编程。以路由器为例，它的最主要任务是分析进入的数据包，并决定从哪一个端口转发它。通过采用一些专用硬件完成细分任务，例如缓冲器管理和加密解密操作等，多个内核可以并发运行，从而获得更高的吞吐能力。

图2所示的是如何通过专用硬件的组合和多核架构的运用来构成高效路由器的基础。

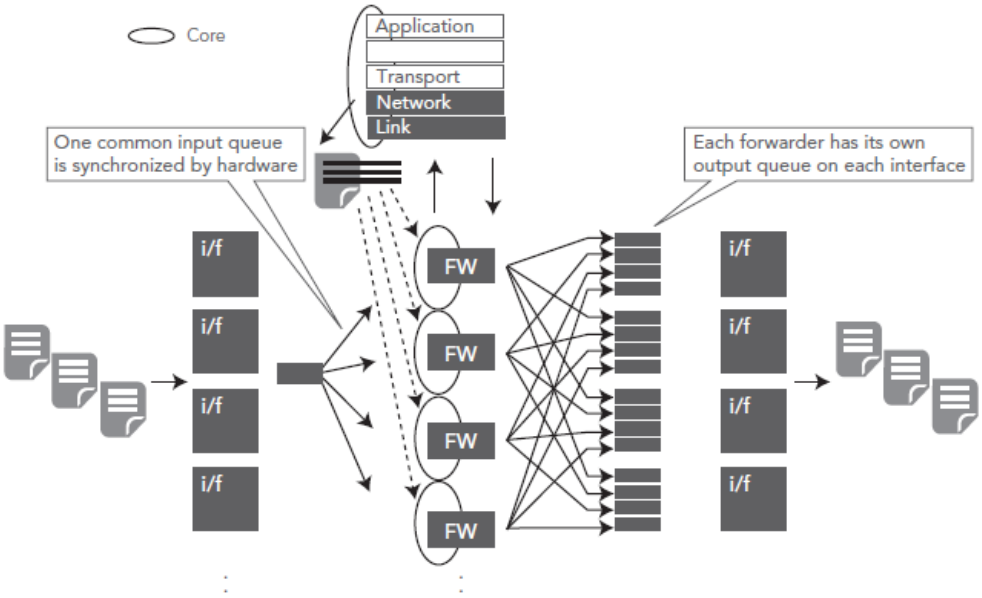

图2：在路由器应用中使用多核技术进行硬件负载分流

将多处理器系统集中为多核架构

很多设备已经通过采用多进程技术来获得更高的性能，或将不同需求的系统进行整合为一体。随着多核处理器的流行，大量设备都开始转向采用更廉价、更低功耗的多核技术。同时我们需要认识到，多核并不是简单地将多个处理器单元聚合在一起。实际上，多核与多处理器有很多重要的区别，将多处理器系统整合为多核架构并不是那么简单。

多处理器系统与多核架构间的一个主要区别是，多处理器系统中CPU之间的分离更加明显和清晰。通常它们之间都有明确定义的总线，即使它们共享外部内存，但也是各自完全独立运行。在多核架构中情况就完全不一样了，其共享程度要高得多。依赖于架构特性，多核系统中的各个内核会有大量的资源共享，例如中断控制器、设备和高速缓存等。这种架构特性在某些情况下是一种特有的优势，最典型的情况就是共享同一个操作系统（SMP）。但当同时运行两个分离的操作系统时，例如一个实时操作系统（如风河VxWorks）和一个通用操作系统（如Linux），这种架构特色却会增加应用难度。当在多核架构中运行多个操作系统时，你可以选择让这些操作系统之间频繁地相互通信，不过更简单的方法是采用Supervisor或者Hypervisor，这是一段用来管理多个操作系统的代码。

嵌入式Hypervisor

Hypervisor（有时也称作supervisor）是用于管理运行于其上的一个或多个操作系统的一段代码，它通过创建运行于其上的分区（或虚拟机）来实现管理。这需要对系统的某些特定方面进行虚拟化。通常有三种类型虚拟化：

1. **CPU：**通过虚拟化CPU，可以在一个物理CPU或内核上运行多个虚拟CPU。这需要对实际CPU进行时间分割（或采用基于优先权的算法），让每个虚拟CPU占用实际处理器时钟周期内的时间片段。
2. **内存：**通过虚拟化内存，可以分隔物理内存，使多个分区都能分别使用实际内存的一部分。运行于该分区上的操作系统同样可以使用虚拟内存来运行进程。在这种应用模式下，

实际上拥有三个级别的内存层次：一个面向Hypervisor；一个面向运行于虚拟机上的操作系统；一个面向运行在进程中的各种应用。

3. **外设：**当多个虚拟机需要共享外设（如串行端口、以太网端口、图形处理等）时，还需要虚拟化各种外部设备。这通常需要在分区中使用明确定义的接口，让运行在其上的操作系统可以进行API调用，而无需直接对物理设备进行访问。完成设备操作的实际代码可以位于两个区域——在Hypervisor上或者客户操作系统上。

在多核架构中，较为普遍的Hypervisor应用模式是，不对单个内核进行多个分区，而是将某一个或多个内核组合起来构成一个虚拟板，这种类型的Hypervisor常常成为Supervisor。在这种情况下，因为不需要对内核进行时间分割，从而使对应的操作系统具有更高的运行效率。更常见的是内存和设备这两个方面进行虚拟化，由此来确保操作系统之间有完整的保护机制来共享设备。

一般来说，Hypervisor有两种类型。类型1是专用Hypervisor，其体积很小并且直接在硬件上运行。类型2通常运行在操作系统之上或与操作系统联合运行，使用宿主操作系统的资源。

最佳的嵌入式Hypervisor是类型1，而大多数服务器Hypervisor都采用类型2。嵌入式Hypervisor（如Wind River Hypervisor）与服务器Hypervisor（如VMware或基于内核的虚拟机）之间的区别是来自于不同的需求。

这些需求推动嵌入式Hypervisor具有了以下的设计方向：

- 设备被直接映射到客操作系统上，以实现最大的性能和最佳的分离。
- Hypervisor是可扩展的，意味着不同实体的虚拟化可以根据对性能和隔离度的要求而任意配置。
- Hypervisor通常体积都很小，这样更易于获得认证，同时也更能确保其快速运行。
- 访客操作系统通常被“超虚拟化”（Paravirtualized），也就是说它们被专门进行过修改，能够在Hypervisor上高效地运行。

依据硬件支持的数量不同，Hypervisor的具体实现有很大的变化。一些现代化处理器在硬件方面采用三级内存系统，可以将Hypervisor对性能的影响降到最小。但通过采用适合级别的“超虚拟化”，Hypervisor也可以在无需特定硬件支持的情况下在处理器上高速运行。

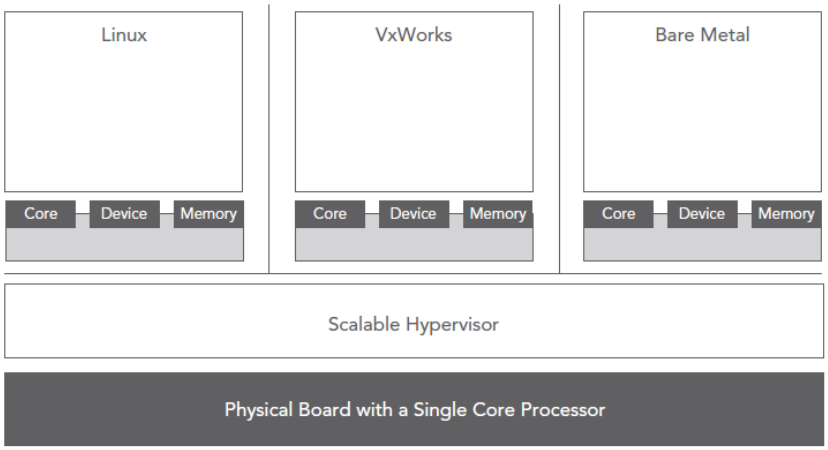

图3：在多虚拟机上使用Hypervisor共享内核

典型应用情景

尽管多核架构和操作系统可能采用众多的组合类型进行配置部署，但还是有一些较为常用的应用情景。
在使用多核系统时，操作系统可以通过两种基础方式来管理内核：一种是对称多处理（SMP），即一个操作系统来控制多内核。当一个内核处于可用状态时，它将根据预定队列运行下一个线程；另一种是非对称多处理（AMP），即每个内核单独运行一个操作系统，这些操作系统可以完全相同或者各不相同，例如有的是RTOS，有的是通用操作系统（如Linux）。

两种方式都具有各自的优缺点，如何选择最佳的方式取决于具体的应用。SMP的优势在于当运行线程数量比处理器数量多时，可以实现内核上的线程均衡。而AMP的优势是对内核间的同步需求较少，因此效率更高。

在SMP系统中，为了获得更可靠的结果和更高的性能，通常会采用“联姻（affinity）”技术将特定的线程和特定的CPU进行绑定。这可以有效降低从一个内核向另一个内核迁移缓存所造成的影响，但是同时也损失了进行线程负载均衡的能力。

AMP通常还结合Supervisor/Hypervisor共同使用，用于处理共享资源以及保护操作系统在出现故障时的相互影响。

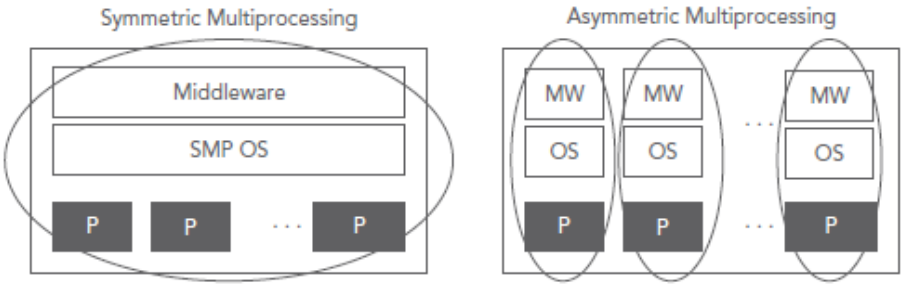

图4：SMP和AMP对比

应用情景1：SMP

SMP可以适用于很多不同的情景。这里我们以网络交换机为例，以便与在同样的设备中采用SMP模式进行对比。在如图5中所示的设备中，利用多核技术的并行性来尽可能地提升千兆交换机的包转发能力。其中使用的操作系统既可以是实时操作系统（如VxWorks），也可以是通用操作系统（如Linux）。通过运用affinity技术（在特定内核中锁定执行线程），位于数据平面的包转发引擎可以完全脱离高速缓存器而运行，利用硬件中特有的负载分流功能来实现缓冲器管理、数据加密解密等等。在本文中“开发工具支持”一节中将会详细描述该系统中的工具（Tools）部分。

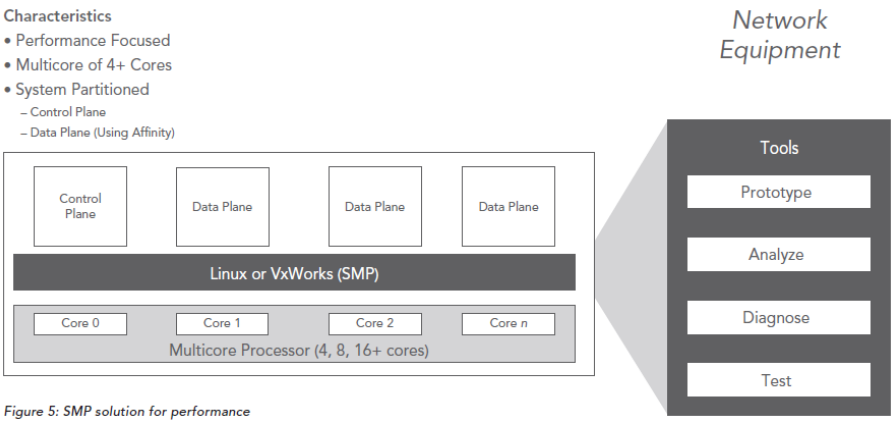

Figure 5: SMP solution for performance

图5：针对性能的SMP解决方案

应用情景2：可监管AMP

图6中所示同样的网络交换机采用了受监管AMP模式（Supervised AMP）。在此系统中，有一个内核专门用于控制平面，运行一个完整的的操作系统。在SMP模式下，如果需要获得控制平面上的更高性能，SMP操作系统可能需要使用多个内核。同时，其他内核也可以用于一些专项的任务，例如包转发等。这些内核上可以运行精简操作系统直接运行执行程序(executive)，从而使交换机的包转发更加高效。“执行程序”将在“开发工具支持”一节中详细描述。

有一个运行于内核之下的Hypervisor用来管理共享设备，例如中断控制器等。在此情景下，Hypervisor只是在需要的时候才活跃起来，例如在安装期间。在此之后，操作系统和执行程序将在它们对应的内核上全速率运行。

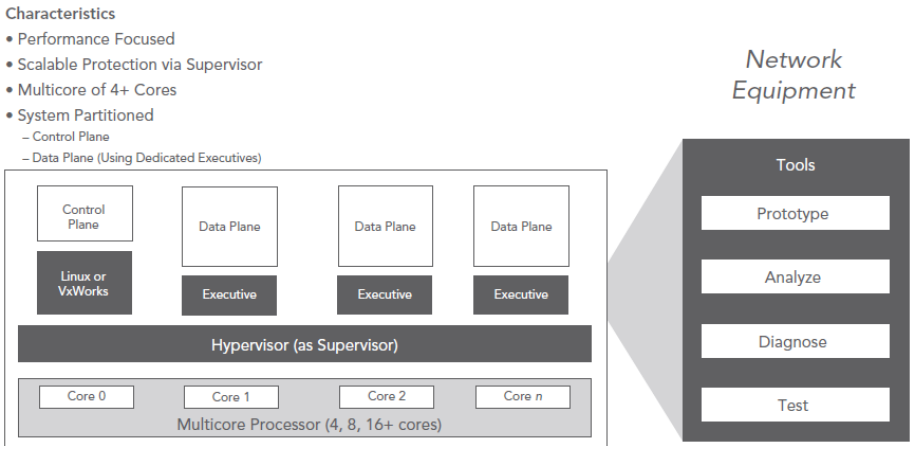

图6：兼顾性能和保护的可监管AMP解决方案

应用情景3：Hypervisor

有些不同的应用情景是针对需要将不同的操作系统与不同的特性混合使用的情况：有可能一方面需要通用操作系统（如Linux）中丰富的功能特性，例如图形处理和连接性等，另一方面又需要操作系统具有Linux所无法达到的实时特性，此外你还有可能很难将已有代码导入到新的操作系统中。还有一种常见的局面是，系统中的某一部分需要达到更高的认证级别。

无论是哪种原因导致你采用了混合的操作系统，Hypervisor都将成为帮助你管理和隔离操作系统分区的高效技术，无论是单核运行还是多核运行。

图7所示的是一种“对等系统”。每个操作系统都不要关心其他操作系统的运行情况，而且在某一操作系统出现崩溃时，其他操作系统在其重启时也能保持正常运行，不受任何影响。

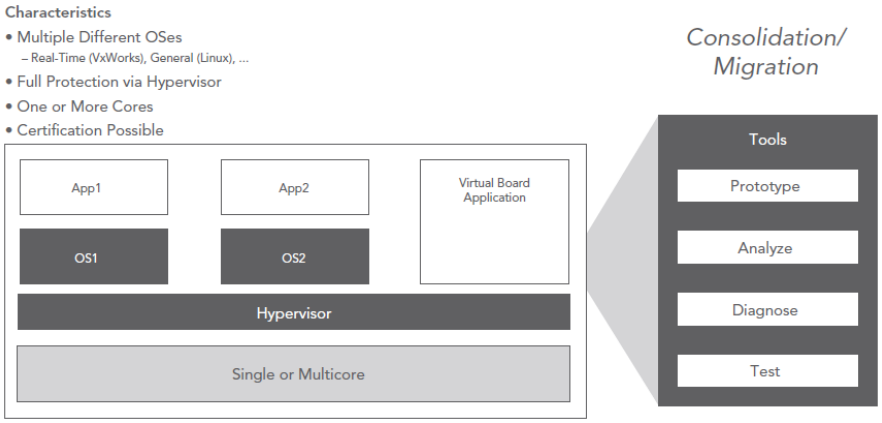

图7：针对整合和保护的Hypervisor解决方案

多核和多操作系统开发人员面临的问题

将代码分割以实现并行化，这只是基础性的挑战。在项目中采用多核技术的开发人员还面临着很多其他的问题，包括：

- 对操作系统配置、资源共享和系统引导的运行时（run-time）支持
- 内核间的通讯（IPC）
- 用于配置、原型化、分析、诊断和测试的开发工具支持

所有这些问题都必须找到答案，以便及时地完成系统开发。本节将进一步详细讨论这些问题。

运行时（run-time）支持和操作系统选择

操作系统的选择及其配置是非常重要的。在很多情况下，操作系统可能由于前期投资或一些特定需求等原因已经被固定了。如果你需要获得更好的实时特性和性能，实时操作系统是不二的选择；如果你需要丰富的应用软件，通用操作系统更符合你的要求。更大的困难在于如何使一个或多个操作系统运行在多核架构下。你究竟需要一个还是多个操作系统？你应该使用SMP模式还是AMP模式？还是需要两种同时使用？在采用NUMA或者硬件多线程的非对称系统中，这种架构具有一定的优势。使用Hypervisor有没有好处？对于特定系统的专用内核应该使用什么操作系统？多核与Hypervisor能够帮助你根据系统的需求，对以上各方面做出正确的选择和组合。

精简/微缩执行程序

前文对SMP、AMP、Hypervisor和Supervisor的优势都已经在做了详细的说明，现在让我们进一步深入到特定任务使用的专用内核方面的细节。当只需要在一个内核上执行某一简单的任务，例如循环轮询（polling loop）时，感觉上或许不需要使用操作系统。这的确是事实，这

完全不需要一个完整的操作系统。你可以采用更轻便的执行程序（Executive），由此提供的API来进行硬件访问和操作就能够满足需求。经常情况下，你可以从芯片厂商那里获得这些执行程序。但是，如果你准备在执行程序之上加入代码，你可能需要得到更多的支持。例如，你期望在代码中植入一个printf()函数语句，并将其重定向到一个映射在你的Linux分区上的串行端口，那么该如何实现呢？另外，调试又怎么做？
有时需要的支持是一个简单的执行程序不足以提供的，这时更好的解决方案是使用一个操作系统。不过，这个操作系统可以缩得很小，其动态尺寸（活跃指令代码）少到可以放进Cache之内，但仍然具有必需的连接性、调试功能和分析工具等。在最好的情况下，你可以得到一个紧密集成的微缩操作系统，还带有从芯片厂商那里得到抽象代码。

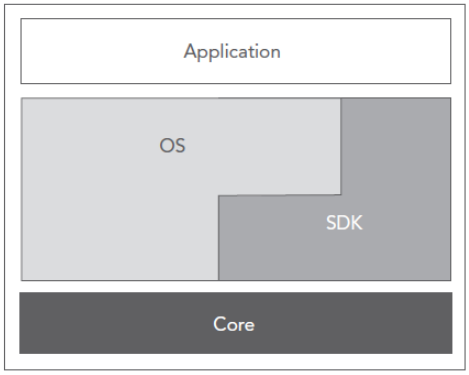

图8：精简/微缩执行程序架构

图8所示的是一个与芯片厂商软件开发套件（SDK）集成的微缩操作系统，在提供常规功能的同时还实现了对硬件的特定操作。

内核与分区通讯

内核间和分区间的高速通讯不仅对应用间的数据交换需求非常重要，也是实现一系列其他功能特性的保障。这些功能包括远程I/O、printf、文件系统、网络通信、系统调试等等。当内核数量超过I/O资源数量时，就需要进行共享。以调试器为例，当需要同时调试多个分区时，通常不能给每个内核分配独立的调试设备。然而，可以通过与某一操作系统实现通讯连接，以代理的方式访问其他分区。

具体需求不同，希望实现的通讯级别也不同。某些时候，内存共享和少量功能就能够满足需求；但有一些时候，可能需要更高级别的消息通讯系统。通过标准化协议和API可以使代码传递更容易。许多进程间通讯（IPC）机制使用了标准套接字接口，使程序变得非常轻便，既可以放在小到几个紧密捆绑而成的小系统，也可以放在更大的采用了TCP/IP标准协议的分布式系统之中。

开发工具支持

在采用多核技术的开发过程中，工具显得更加重要。多核系统的复杂性导致采用传统的开发手段变得非常困难。简单的“编辑/编译/调试”开发周期已经不能满足需求。系统的配置、原型化、分析、诊断和测试等方面面临的挑战迫使开发人员采用更高级别的工具支持。

原型化与仿真

在某种工具上进行仿真以取代在真实硬件上运行操作变得越来越流行，其原因有很多。有的是因为真实硬件还没有具备条件，或者由于太昂贵而不能配备给每一位开发人员。有的是因为需要在特定的方式下进行系统调试，而这种方式又是使用真实硬件所不能实现的。最常见的原因是对于大多数软件开发人员而言，在不需要考虑实际硬件问题的情况下进行工作会更加容易，可以在每次调试中更加关注特定的变量。在开发初期就进行持续的测试，而不是到了开发末期才开始测试，这种开发流程的革新已经变得越来越重要，而仿真器在其中扮演了非常关键的角色。

仿真器有多种类型，用于满足各类情况下的需求。一种类型是快速仿真器，能够模仿操作系统行为的快速功能性，可满足大多数要求。通过使用开发所基于的主机上的各种指令集，可以让应用在其上快速地执行。

另一种类型是指令集仿真器（ISS）。在多核开发过程中，使用指令集仿真器有很多优势。首要的优势就是可以不需硬件条件就绪即可完成调试，除此之外ISS还具备多种属性来提供强大的多核调试功能、准确的程序执行和高效的系统仿真等。

ISS在进行系统仿真时能够多次以完全同样的方式重现代码执行过程，这在真实硬件环境下是无法实现的，真实硬件环境中总会存在时序偏差，导致软件的行为改变，因此很难完全重现同样的执行过程。能够提前预测时序问题的出现，由此带来的好处是使调试变得更加容易，因为问题每次都是以同样的面目呈现出来。通过系统仿真，你可以模仿整个操作系统行为并且随时中断所有的处理器，停下来观察运行中的情况。这些功能特性对于多核调试是非常有用的。

有些仿真器还有更高级的功能，例如软件追踪功能和逐步回退。尤其是逐步回退功能，如果在任何情况下都可以前进和回退，以便观察发生的变化，这对很多状况真的非常有帮助，特别是在调试设备交互访问的时候更是如此。

诊断与调试

在进行多核开发时的另一艰巨任务是调试。与普通的顺序代码相比，调试互动线程的困难程度将会增加十倍以上。这是因为必须考虑到另一线程或处理器可能在任何时间点也对共享数据进行操作。突然之间，调试人员不得不用全新的眼光来看待每一行代码。

死锁和竞争状况是两种最常见的多线程软件Bug。死锁的发生是由于两个或两个以上的线程每个都占用信号量（taken a semaphore），导致发生相互等待而无法继续运行的现象。竞争状况使由于多个线程以交叉存取的方式同时访问同一个全局数据，导致不可预测的结果。经常出现的情况是，解决一个死锁问题却引发了一个竞争状况问题，反之亦然。

解决竞争状况问题的难点在于它们依赖于时序，它们只会在非常特殊的情况下才会发生。由于每次运行的时序都不尽相同，每次运行程序竞争状况出现的征兆也都不一样。如图9所示，可能需要进行上百万次循环，才能遇到一次导致数据损坏甚至整个应用情景和系统崩溃这类严重的问题。

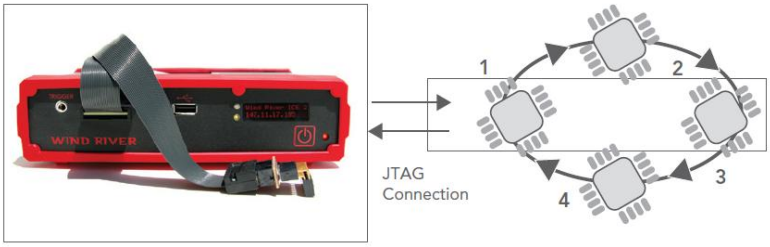

图9：通过JTAG连接的片上调试

在这些情况下，运行时（run-time）分析工具非常有用。它的典型运作方式是在事件发生时以日志的形式记录下来，从而让我们可以在题产生之后作为详细分析的依据。还可以向其中增加自己的用户事件（user event），从而可以观察与其他事件相关联的特定部分软件何时会被执行。通过在代码中加入操控共享数据的用户事件，可以观察到有没有发生数据在没有受到互斥体（mutex）良好保护的情况下被访问。

通过加入代码来检查时序问题是非常乏味的，你必须不断地重编译和重启动应用，并记得在之后移除他们。优秀的调试工具允许动态地向正在运行的系统中加入代码，这与加入调试断点的概念很相似。

这种调试模式可以不必停止系统运行就可以观察分析各种问题，这对于很多设备开发是非常有利的。但有时我们真正需要的是看到硬件中正在发生的情况。这里还介绍了其他一些工具，只不过它们并不那么完美，因为可能会插入一些测试性代码或者只是对真实硬件的模拟，所以会对系统行为造成些许的改变。当今的大多处理器都具有片上调试（on-chip debugging）功能。片上调试通常要靠芯片板上的一些针脚（如JTAG）来实现，可以让片上调试器直接控制处理器，而不再需要使用软件代理。在多核架构下，调试器能够通过一个名为“扫描链”的独立命令序列来控制所有的处理器和设备。

现在的片上调试可以在同一扫描链中同时控制（启动、停止等）多个处理器，并且只需很短的潜变时间（指从一个内核响应直到所有内核都响应的时间）。通过这一重要特色，可以确保当一个内核被停止时，其他内核不会失控和改变状态（或产生缓冲器溢出）。此外，你可以通过一个集成开发环境对内核进行控制，直观地观察各个内核的情况。

立即停止所有内核的功能使片上调试开发器具备了强大的系统级调试权力，可以在必要时候停止整个运行系统。

除了运行时分析工具和片上调试器外，传统基于代理的调试器也仍旧被保留使用。特别是用于调试分析那些不需要中断系统其他部分也可以停止其运行的代码，使用传统方式更为有效。

分析

开发人员刚开始使用多核技术时的一个普遍问题就是无法达到所期望的性能。众所周知，多核系统的性能的确非常难以预测、分析和推断。系统应用不能获得高性能有很多原因，包括同步问题（例如等候信号量）、缓存问题、内存或I/O-bound应用等等。在没有真实数据的

情况下进行推测是非常困难的。分析工具可以帮助判断造成系统性能降低的瓶颈所在。

剖析器（profiler）是推断代码效率低下原因的一种高效工具。通过使用层级化剖析器（hierarchical profiler），可以显示在一个完整的调用树（call-trees）中耗费了多少时间，而单层剖析器（flat profiler）只能呈现某一独立功能所耗费的时间。

有了剖析器所提供的有用信息，就能以此为基础来决定从哪些方面着手开始提升并行处理能力。

为了进一步了解多核应用的内部运行状况，还需要使用能够显示事件时间序列的运行时分析工具。最终的输出结果大致如图11的截屏所示。

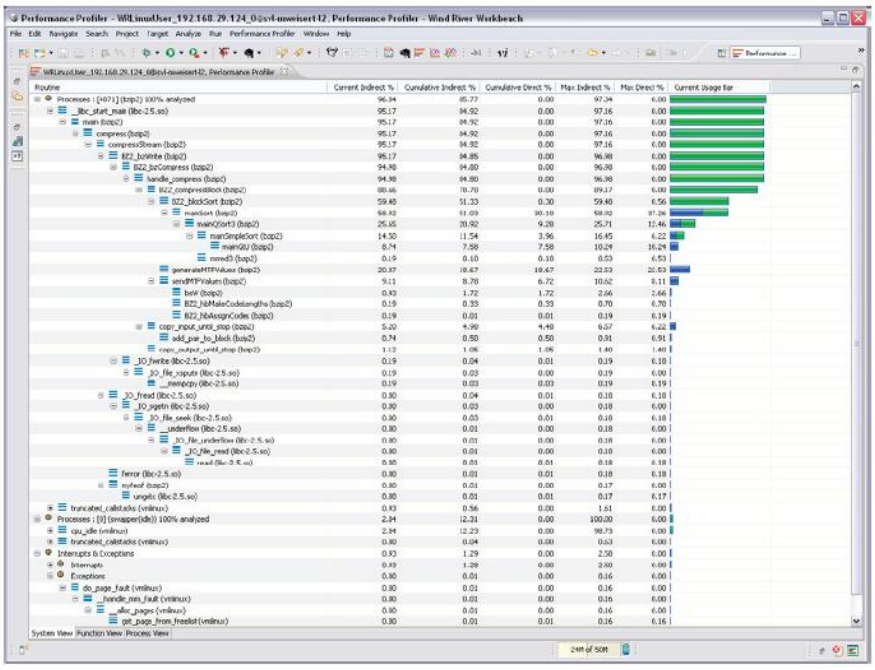

图10：Wind River Workbench等级化剖析器功能

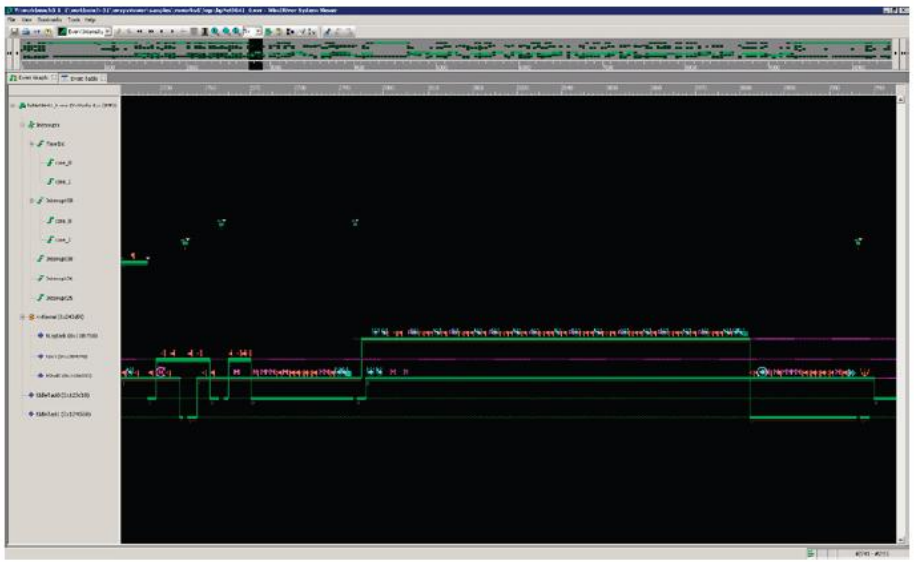

图11：Wind River Workbench系统查看器（System Viewer）功能

结论

多核如影随形！而且真真切切！为了尽快利用全新多核处理器带来的优势，最佳的捷径就是采用商用化的工具和运行时系统，而不是别人都在开跑车了，我们还在自己埋头重新“发明”车轮。多核开发与单核开发所用的方法截然不同。

并不存在单一的解决方案能够解决多核应用的全部问题，也不存在能够把单核程序转换成多核程序的法宝。我们需要的就是一套运行时配置和工具，让自己的应用高效运行起来。我们需要这些工具和运行时很好地集成起来，不仅自成一体，而且与我们所选择的芯片硬件也天衣无缝。只有这样，我们的应用才能充分发挥多核所带来的优势。

Wind River 就在您身边

北京代表处	北京市朝阳区望京中环南路9号望京大厦B座18层	邮编：100102	电话：010-84777100	传真：010-64398189
上海代表处	上海市西藏路585号新金桥广场3-H,I,J室	邮编：200003	电话：021-63585586/87/89/90	传真：021-63585591
深圳代表处	深圳市福田区车公庙天安数码时代大厦A座606室	邮编：518040	电话：0755-25333408/3418/4508/4518	传真：0755-25334318
西安代表处	西安市高新区科技二路68号西安软件园秦风阁H103	邮编：710075	电话：029-87607208	传真：029-87607209
成都代表处	成都市高新区天府软件园二期D7 14层	邮编：610041	电话：028-65318000	传真：028-65319983

关于风河更多内容请访问：http://www.windriver.com.cn　　Email：inquiries-ap-china@windriver.com

WIND RIVER