WIND RIVER

针对新数据面的多核网络

概述

基于 Linux 的风河网络加速平台（Wind River Network Acceleration Platform），可以加速各种数据面网络应用程序，包括路由，网络安全、无线基础设施和其他 Linux 平台上需要高性能网络的应用程序。这种现成的解决方案带有完整的多核开发工具套件，并提供了必要的基础设施，使客户能够方便、快捷地开发出在多核处理器上运行的应用程序，与仅使用传统的对称多处理（SMP）方法相比，可以实现更高的性能和伸缩性。

Intel 整合了 Intel DPDK 及 QuickAssist 技术的英特尔架构（IA）处理器，可以创建出高性能的数据面应用程序，而这在以前则需要使用针对特殊用途专门开发的处理器或客户定制专用集成电路（ASIC）才能实现。最近的一些创新成果简化了英特尔架构的数据路径，使处理器核心能够以前所未有的高效率传输和接收帧，从而针对从 SOHO 应用到大型企业级设备在内等各种设备创建出经济有效的高速网络解决方案。

目 录

用于网络设备的多核技术

多核技术和其他技术的进展使网络设备能够前所未有的支持更复杂的处理，而无需借助特殊不易扩展的专用集成电路。由于不再受限于轻触型（Light-touch）操作，现在网络应用程序可以包含更高层的处理、高级安全、以及通常由终端系统执行的工作，例如对包内容的处理。

这些新特性使得控制面和数据面之间传统的区分变得模糊起来，原来与控制面相关的工作现在也开始要求数据面的性能。多核处理器的高性能和多功能性造成了这种环境上的变化，促使人们开始重新思考处理器的角色，并催生了处理器架构的变化。原来使用的处理器架构在高技术个性化（High-touch）应用和高吞吐量应用间是不同的，而目前则趋向于使用更加简单的模式，即对于这两种应用都采用单处理器架构。

除了使产品能够在数据路径中执行更多的工作外，这种转向单一平台的趋势还有很多其他好处。单一的平台可以节约成本、提高效率并增加代码的重用率，从而提高质量、加速上市时间、并且降低软件和硬件的维护成本。

数据面处理的独特挑战

我们正处在一个飞速连接的时代，在下一个数据包到达之前，只有几纳秒的时间供系统对现有数据包进行处理。因此，硬件和软件必须要协同合作，才能完成网络应用程序要求的任务。硬件需要以极高的吞吐率和尽可能短的等待时间来接收数据包并将其放入预先分配的内存位置。而软件则必须检测到这些数据包的到达，并以较低的系统开销来执行这一任务，以便使大多数 CPU 周期能够用于真正的工作。

过去，网络设备注重于提供高性能的轻触型处理能力，例如在桥接和基本数据包转发应用中的处理。这种类型的工作更加偏向于网络 I/O 而不是原始处理能力，更加强调数据路径的效率而不是处理器的复杂性。这造成了对利基专用数据面处理器和相关微内核的要求，这样的处理器在网络 I/O 方面的性能非常出色，但是对于更高层次的工作则缺乏灵活性。这些处理器要求不同的编程范式，这种范式需要专有编程语言的支持，并且要求了解任务分配的底层知识和详细的周期预算。为了使用这些专有的平台，设备制造商往往不得不对他们的软件开发人员进行重新培训。此外，生成的代码还往往需要精细的调试和校正，而这极大影响了软件适度扩展的能力，因为之后添加的功能可能会打乱这种精细的平衡，因此每次发布新的版本时都需要额外的代码重新优化周期。

与之相反，控制面的处理更加偏向于高 CPU 时钟频率而不是网络 I/O 的效率，而这得到了当前通用处理器很好的支持。这些处理器通常连接到通用 PCI 总线上的网络接口卡（NIC），这种方式虽然比前一种技术有所改进，但是在传输小数据单位的时候则效率不高。因此，在控制面处理中表现良好的通用处理器并不一定适合数据面的使用，这就是处理器角色的二分性。

从软件的角度看，有很多与数据包的发送和接收相关的隐藏成本，尤其是对于 Linux 等通用操作系统，因为这些操作系统必须公平地对待网络应用程序和非网络应用程序。即使在今天，虽然在网络 I/O 方面取得了很多进展，但是 Linux SMP 的性能依然非常一般，而且扩展性能不佳。不过，对于很多软件开发者而言，Linux 因其灵活性及通晓性，依然是一个非常具有吸引力的平台。因此，虽然 Linux SMP 或许不能提供最佳的性能，但是任何替代解决方案也必须能够与 Linux 一起工作，在该范式的限制条件下加速应用程序。

多核软件的两难选择

随着 CPU 核心数量的增加，传统的对称多处理系统表现出很差的扩展性。SMP 是一种共享架构，所有的核心都运行一个操作系统实例，并共享所有的数据。如果设计合理，那么如今的应用程序可以不太费力地移植到可感知 SMP 的通用操作系统中。虽然经过精细调整后，这些应用程序可以实现很好的性能，但是缺乏扩展性则成为了一个问题。随着核心数量和线程的增加，共享数据结构中出现互锁的频率也会增加，再加上其他性能瓶颈，从而造成共享内存模式的崩溃。这就出现了包处理性能的平整现象，即随着核心数量的增加，性能只有微小的改进，在一些情况下甚至还会出现性能的下降。

尽管如此，很多系统架构师还是决定忽视这些缺点并选择传统的 SMP 方式，因为这种方式不仅使用方便，还可以缩短产品上市时间。在多核环境下设计软件时，在开发的便捷性和性能之间如何取舍一直是一个两难选择。要实现最佳的性能并真正利用多核处理系统的优势，系统架构师们需要舍弃传统的 SMP 方式而采用非对称模式。这种模式更加适合数据路径的处理，但是往往要对现有的软件代码进行大量的改写。

英特尔架构处理器的演进

英特尔架构的处理器经常被用于要求大量处理的应用中，它们凭借具有很高的时钟频率以及大容量高速缓存的明显优势，对于包内容进行处理的应用程序有着极大地吸引力。这种处理操作的代码通常有着数千条指令，频繁地访问大数据结构，并不断地对包数据进行操作。由于核心数量多、CPU 时钟频率快，因此在一定的时间预算内就可以执行更多的指令，同时大容量高速缓存也减少了对外部存储器的访问。

最近的处理器改进聚焦于在固定的功率预算下提供更高的效率、创建更加高效的指令流水线、更高的指令执行并行性、以及更加有效的算法，这些改进使每个周期内可以执行更多的指令，从而在可用的 CPU 周期内执行更多的指令。随着英特尔超线程技术的重新引进，单一的核心可以同时运行多个硬件线程，如果当前运行的线程被阻塞，那么就切换到其他线程，从而避免了与内存访问相关的等待时间。这些改进使处理器能够在现实世界的应用程序中提供更优越的性能。

此外，随着市场倾向于更加智能化的数据面，英特尔架构的处理器也发生了演化，通过将大容量高速缓存、效率的改进以及先进的总线技术、CPU 内存和网卡之间更加直接的路径等技术相结合，使处理器架构可以支持高技术个性化和高吞吐量的应用。针对网络接口卡的连通性，PCI Express 替代了内存映射的 PCI 总线。共享前端总线（FSB）的效率不断优化，之后则被点对点、低等待时间的英特尔快速通道互联（QuickPath Intelconnect）所替代。内存控制器变得更宽，以支持更高的内存带宽，随后的嵌入集成更是大大缩短了与内存之间的路径。非均匀访存模型（NUMA）的支持引入了内存距离的概念，为软件提供了在本地存储器和远程存储器之间进行选择所需的信息和灵活性。

这些改进使网络接口卡现在能够以前所未有的高速度和高效率将接收到的帧存入到本地内存。此外，再加上英特尔架构处理器一贯的高 CPU 速度和大容量高速缓存特性，因此英特尔架构的多核平台非常适合大吞吐量和高技术特性化的应用，能够实现市场领先的包处理能力，并且能够加速密集数据路径的工作，例如加密、压缩、以及新数据面中的深度包检验。

英特尔数据面开发工具套件

英特尔数据面开发工具套件（Intel DPDK）针对高速数据 I/O 应用而设计，是一个数据面工具库，能够优化数据路径，创建专门用途的用户空间应用程序，并提供本地 Linux 所无法提供的性能伸缩性。Linux 等通用操作系统必须经常平衡各方利益，并且需要包含一般的安全机制，以防止应用程序相互之间或者与内核发生干涉。这虽然在通用环境中是必要的，但是在专门用途的网络应用程序中则造成了额外的负荷。

与之相反，Intel DPDK（图 1）提供了传统 Linux 系统调用的替换方案，提供了一套可以从 Linux 用户空间调用的 API。Intel DPDK 在整个英特尔架构的产品线中得到支持，为从英特尔凌动（Atom）到英特尔至强（Xeon）的各种处理器提供了统一的编程范式。这些库在初始化的时候设置，并运行于 Linux 内核之上，为应用程序提供了常见的实用程序，而且在调用时不涉及 Linux 内核，从而减少了数据路径中用户-内核交互作用的开销。

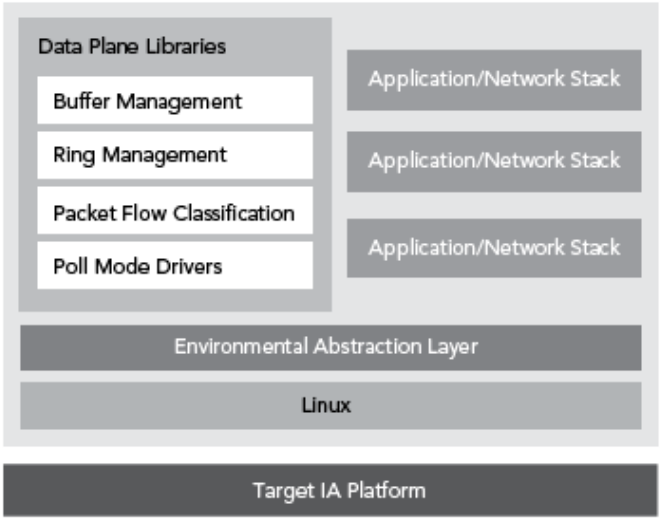


图 1：Intel DPDK

应用程序位于用户空间中，并使用数据面库来接收和发送数据包，在进行常规数据面处理时绕过了 Linux 内核。Linux 内核对 Intel DPDK 应用程序的处理和其他用户空间应用程序一样；它的编译、链接和载入都是以常规方式进行的。该应用程序作为一个单一的 Linux 进程启动，使用 Pthread 线程实现并发处理，并通过 Pthread 亲和性将每个线程附着到指定的内核。

Intel DPDK 针对专门用途的数据 I/O 密集型网络应用程序而设计。这些库可以帮助应用程序有效地接收和发送数据，并为更复杂网络软件的开发提供基础构件。这种方式使软件开发人员可以关注于应用程序本身，而不必花费精力来确定如何配置和精细调节操作系统以提高性能。与此同时，虽然这些库的应用是针对英特尔架构进行优化的，但 Intel DPDK 并没有引入或向应用程序强加任何特别的硬件范式。相反，可以将 Intel DPDK 视为一个工具包，应用程序开发人员可以利用它来提取、提供实时软件开发人员所熟悉的性能增强技术的优化实现，例如轮询或使用无锁环路和零拷贝缓冲，所有这些都可以在用户空间中提供，同时避免了内核的常见问题。

风河网络加速平台

风河网络加速平台是一个完整的多核解决方案，将加速的数据面和标准的 Linux 相结合。加速的数据面包含一个或多个核心，每个数据面核心会调用一个网络加速引擎（NAE）。网络加速引擎提供了高性能的网络数据路径，替代了 Linux 内核和堆栈进行数据面处理。当部署在英特尔平台上时，网络加速平台会利用 Intel DPDK 在数据面上进行英特尔架构特定的优化，同时使用 Linux 进行管理控制。Intel DPDK 提供了与硬件打交道的功能来进行底层处理和数据包 I/O，而网络加速平台则在 Intel DPDK 上方提供了优化的网络堆栈应用和应用程序支持基础设施。网络加速平台的网络堆栈是完整的，并不限于主路径用例或静态路由和寻址。与 Linux SMP 解决方案相比，网络加速平台可以减少共享内存争用并降低对 Linux 内核的依赖性，因此具有更好的伸缩性。

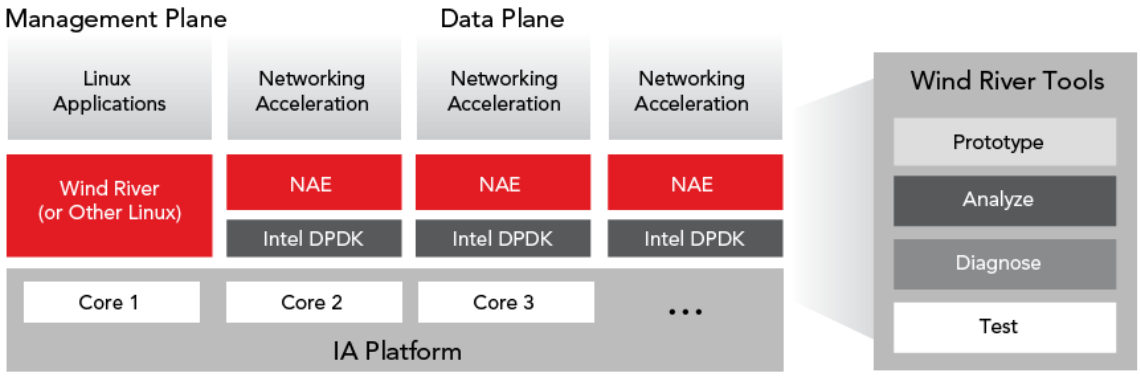


图 2：风河网络加速平台

网络加速平台（图 2）可以加速原生 Linux 应用程序，或者在付出一些额外工作的前提下提高性能，加速运行在网络加速引擎上基于 Socket 的应用程序。网络加速引擎堆栈是高度模块化的，并且针对单线程环境进行了优化，可以加速网络层服务和功能，例如包转发、IPsec、IP 分片和重组、虚拟局域网（VLAN）、以及要求完全 TCP/UDP/SCTP 协议终结的应用程序。可以仅根据需要的组件对网络加速引擎堆栈进行配置和构建，因此既可以生成完整功能的堆栈，也可以生成需要的精简型堆栈。此外还提供了相关的风河工具，可以用于开发、运行时分析、对管理面和数据面上的软件进行调试，此外还包含了用于开发单线程应用程序的工具。还可使用提供的内核模块和管理应用程序，从 Linux 核心对堆栈进行管理。

软件开发人员可以直接在堆栈上使用熟悉的 Socket API 来创建应用程序。网络加速引擎支持用于协议终结的流套接字和数据报套接字，以及允许应用程序访问网络上到达原始数据包的原始套接字。所有的套接字都是零复制的，这意味着当数据从网络堆栈到应用程序以及从应用程序到堆栈时，没有复制操作发生。

开源 Linux 应用程序的网络加速

开源 Linux 应用程序的加速有着非常严格的限制条件，无法对应用程序代码进行任何变更，甚至无法对应用程序进行重新编译，因此很难对其进行支持。这些应用程序针对 Linux 环境的运行而编译，使用了常规的 Linux 机制进行引导、系统调用、并与 Linux 内核交互来发送和接收数据。虽然很多人可能会认同这些应用程序的加速是有好处的，但是要将这些应用程序移植到网络加速引擎的单线程环境中却并不一定可行或令人满意的。

为了满足这一需求，风河网络加速平台对特定的开源 Linux 应用程序进行了加速，方法是截取系统调用，使这些应用程序能够使用 Linux 内核之外的加速 NAE 网络堆栈。这是通过风河提供的套接字中介层来实现的，这一中介层将指定的系统调用转移给本地网络加速引擎，而应用程序和 Linux 内核均不会意识到这一点（图 3）。例如，进行套接字读取时，本地网络加速引擎会将到达的网络数据直接放置到应用程序套接字缓冲区中，而在进行套接字写入时，本地网络加速引擎会将应用程序缓冲区中的内容包格式化后发送到网络中，这些都无需 Linux 内核的介入。这样就实现了较少的上下文切换、更少的数据复制、并提高了性能，而且只需对 Linux 应用程序进行很少的更改或者无需修改。

风河仅针对特定的开源 Linux 应用程序支持这种类型的加速。要获得所支持应用程序的完整列表，请与风河公司联系。

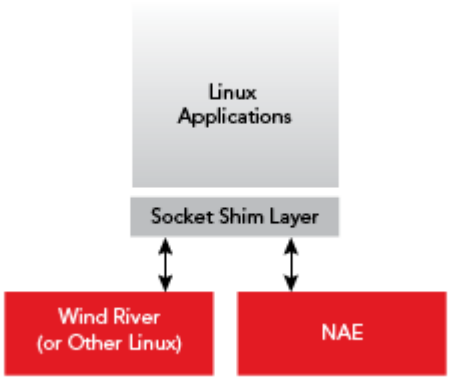


图 3：Socket 中介层

应用示例：下一代防火墙

网络安全提供了一个良好的例子，可以展示应用程序为了利用多核技术的优势是如何变化的。它已经从基本防火墙中简单的包过滤发展成为更加高级的应用程序，可以执行入侵检测和防御（IPS）、网络杀毒、IPsec、SSL、应用程序控制等。这些功能都驻留在数据面，并且要求对数据流进行深度包检验、加密和压缩能力、以及对包内容的密集处理。

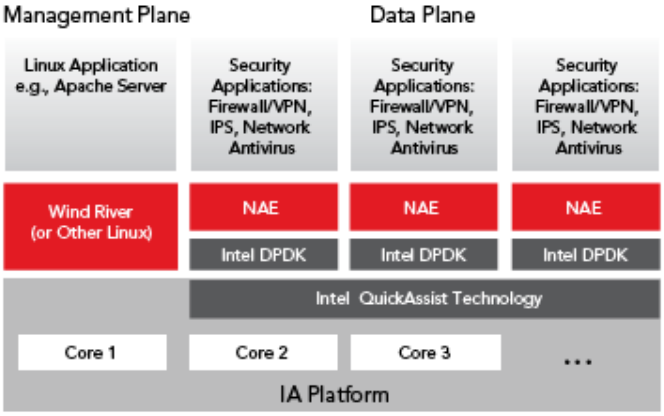


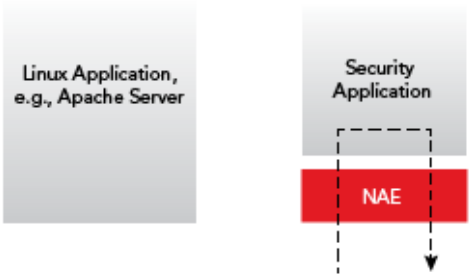
图 4：下一代防火墙

最新的英特尔架构平台带有集成的内存和 PCI Express 控制器，为高吞吐量和低等待时间的数据处理解决方案提供了可伸缩的解决方案。网络接口卡和处理器内存之间优化的数据路径使数据包能够以更短的时间送入到内存以及从内存中向外传输，而高时钟频率和大容量高速缓存则为应用程序在CPU周期预算内执行所需的工作提供了必要的环境。通过英特尔 QuickAssist 技术，软件可以利用硬件卸载来进行 CPU 密集型的加密操作，从而释放处理器供其他任务使用。

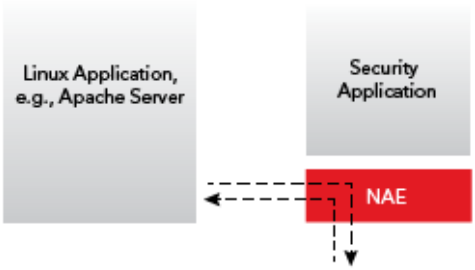
图4显示了一个使用风河网络加速平台和Intel DPDK以及英特尔 QuickAssist 技术的下一代防火墙。英特尔架构的处理器提供了有效接收和传输网络流量的硬件，以及执行这些数据密集型应用程序代码所需的高CPU时钟频率和大容量高速缓存。Intel DPDK 提供了一种机制，可以支持用来替换 Linux 系统调用的高性能方案，从而绕过 Linux 内核的一般问题。最后，在这一英特尔基础设施上还建立了风河网络加速平台，为 Apache 服务器等开源 Linux 应用程序提供加速，并且可以为移植到网络加速引擎的安全应用程序提供更高的加速性能。

防火墙/VPN、IPS 和网络杀毒应用程序都在网络加速引擎上运行，以实现最佳的性能。当数据包被接收到内存中后，Intel DPDK 提供一个无中断的机制，由网络加速引擎对数据包进行处理。如果数据包的目的地是本地 Socket，那么网络加速引擎在自己的网络堆栈中将数据包向上传递，一直到达拥有该 Socket 的安全应用程序，并在必要的情况下使用 Intel DPDK 库。安全应用程序可能会查看进入的数据包，对内容进行分析，以保证数据包中没有恶意软件或其他攻击，然后将数据包发送到适当的接口。这些都是使用零复制缓存和无锁环路在 Linux 内核之外进行的，减少了代价昂贵的复制操作和共享内存争用。如果数据包的目的地是 Linux 中运行的被加速的应用程序（例如 Apache 服务器），那么网络加速引擎会将数据包通过套接字中介层转发给应用程序，并绕过 Linux 内核。

图 5 显示了这两种加速数据路径。



A：网络加速引擎安全应用程序的加速数据面



B：Linux 应用程序的加速数据面

图 5：网络加速引擎和 Linux 的加速数据面

结论

风河网络加速平台是一个高速的商业现货网络解决方案，为应用程序更好提供了可善用多核处理器优势的必要基础设施。与使用传统 Linux SMP 的解决方案相比，其性能要高很多倍，并且提供了一个不受 Linux GPL 许可证限制的数据面环境。当被部署在英特尔架构平台之上时 ,风河网络加速平台可以充分利用 Intel DPDK 和英特尔 QuickAssist 技术，为希望利用英特尔架构处理器优势的应用程序进行加速，并且为包括下一代防火墙、无线基础设施和其他内容处理的数据面应用程序提供加速。

设备制造商们不必再在使用 Linux SMP 的便捷性和专有非对称解决方案的高性能之间进行权衡。网络加速平台可以加速原始 Linux 应用程序，并且可以支持熟悉的基于套接字的编程范式，在一个优化、精简的产品包中同时提供了高性能和便捷，并且提供了在多核环境下开发应用程序所需的所有工具，使软件开发人员可以更加关注自己的应用程序，而不必在底层基础设施的优化上耗费时间。

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

同步关注风河新浪官方微博，关注 **@风河系统公司**

WIND RIVER

风河（Wind River）公司是Intel(NASDAQ: INTC)的全资子公司，也是全球领先的嵌入式和移动软件提供商。从1981开始至今，风河公司一直是嵌入式设备中计算技术的先锋。在当今世界中，已经有超过10亿台产品应用了风河公司的技术成果。公司网站 www.windriver.com 和 http://www.windriver.com.cn/

风河系统有限公司2012版权所有。风河标识是风河系统有限公司的商标，风河和VxWorks是风河系统有限公司的注册商标。本文中使用的其他标记属于其各自的所有者。更多信息请参见www.windriver.com/company/terms/trademark.html。 2010年1月修订