GALLAGHER PR

Exclusive Three-part Series to *The Data Center Journal*
Placed and Written by Gallagher PR

For Jeffrey R. Clark, Ph.D., Editor, The Data Center Journal
jclark@datacenterjournal.com
http://www.datacenterjournal.com/

Improving Price/Performance with Intelligent Server Adapters (Part 1 of 3)

Host-based networking includes implementation of networking functions such as network virtualization, security, load balancing and telemetry. Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) build on the foundation of host-based networking and are intended to operate on commercial off-the-shelf (COTS) hardware equipped with general-purpose x86 CPUs and network interface cards (NICs). But will such a configuration result in an efficient implementation and peak price/performance in the servers being used as compute or service nodes? Extensive testing provides a definitive answer: No.

General-purpose CPUs are simply not designed for high-performance host-based networking, and this results in an extraordinarily high percentage of CPU cycles (and cores) being consumed by networking tasks that could be offloaded to an intelligent server adapter. In critical use cases, the use of such an intelligent server adapter can deliver more than a 20X improvement in overall price/performance.

This three-part series takes an in-depth look at the promise and pitfalls of using general-purpose CPUs in host-based networking applications, and explores the different technologies being utilized in intelligent server adapters.

The Proliferation of Host-based Networking

Host-based networking has garnered considerable interest in the industry. It has been widely deployed in large cloud data centers to enable significant efficiency gains and increase the pace of innovation. Sophisticated networking functions such as virtualization, security, load balancing and telemetry have been moved away from networking infrastructure equipment, such as switches and purpose-built appliances, and have been implemented using software on COTS servers. The virtual switch data path, virtual networking functions (VNFs) and other operating system data path elements such as IP Tables are examples of these software applications. The advantages of this approach are two-fold. First, it simplifies the networking infrastructure, which is now used for simple plumbing between servers, while more complex functionality is implemented at the endpoints leading to more efficient scaling. Second, use of software-based data paths is analogous to SDN, enabling rapid innovation and control over new feature rollouts.

Indeed, the current bellwethers of host-based networking—namely the large data center operators—have long endeavored to evolve networking where it is the easiest and most natural: in the server where operators have full control of the networking stack. It makes perfect sense, of course, to evolve networking close to where the applications that use networking actually run.

With the increased use of virtual machines (VMs) in cloud infrastructures, the virtual switch that sits close to the VMs in the server has become a focus for innovation. More specifically, the virtual switch data path that processes traffic or packets destined to applications in VMs has become an area where host-based networking has seen most of its success to date. Figure 1 depicts where such a virtual switch data path is located in x86-based servers. The opportunities that exist here get to the essence of SDN and NFV and their promise to afford enormous savings in both operational and capital expenditures in data centers worldwide.

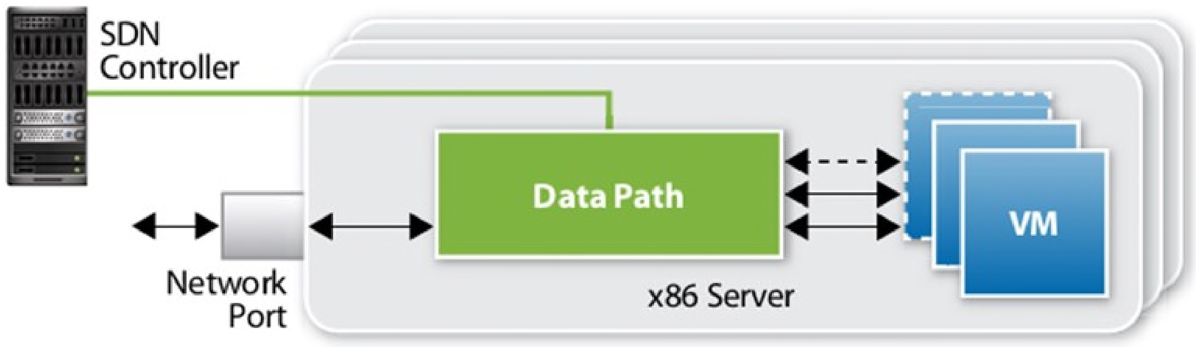

Figure 1: Host-based networking utilizing the virtual switch data path

Because the virtual switch data path is implemented in software and runs in the familiar and ubiquitous x86-based CPU environment, data center operators have been able to freely add new features and quickly roll these out in production deployments. One such new feature involves tunneling for multi-tenant network virtualization, which also evolved initially inside large data centers in various forms and has only recently been standardized across the industry with specifications like Virtual Extensible Local Area Networking (VXLAN) and Network Virtualization using Generic Routing Encapsulation (NVGRE).

Other recent innovations in both proprietary and open source virtual switch domains (e.g. Open vSwitch) involve new tunneling technologies for network virtualization and traffic engineering, as well as faster, more scalable processing of match-action rules used to implement the policies that govern access for applications and users. Future enhancements may encompass stateful security using Linux connection tracking, sophisticated load balancing or flow tracking to provide cut-through or accelerated processing of the data path, as shown in Figure 2.

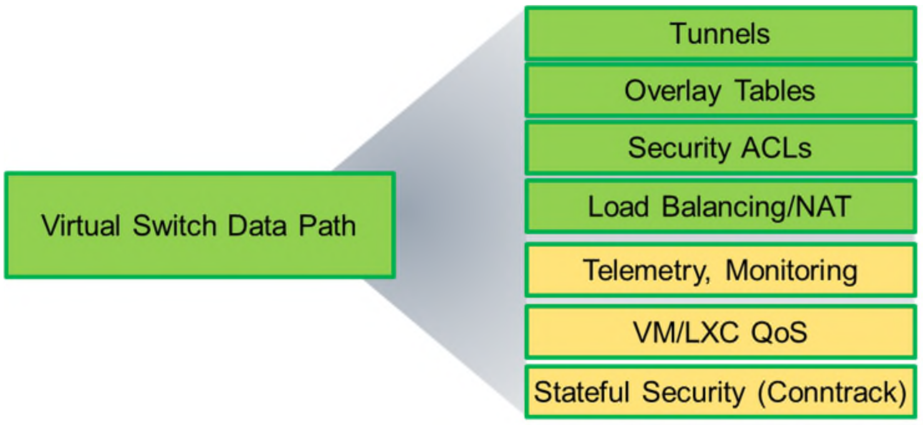

Figure 2: Evolving the virtual switch data path with new features

Besides the virtual switch, other forms of host-based networking are evolving, for example Layer 3-based approaches such as virtual router with Juniper's open source Contrail solution, or the open source Project Calico solution.

Implications of Host-based Networking

The need to implement sophisticated network-related processing in software on the host requires the dedication of a significant amount of x86 CPU cycles. This, of course, results in a reduction of the CPU cycles that are available for VMs, VNFs and other applications. The demand on CPU cycles required for network processing are increasing dramatically with the relentless increase in data rates, such as the transition from 10GbE connectivity in servers today to 25, 40 and 50GbE connectivity, and the increased numbers of tenants and VMs per server that require processing of many more tunnels, services and security policies. The situation will only be exacerbated with the deployment of containers instead of VMs.

Consider the relatively simple example of processing a network virtualization tunnel endpoint with a basic match-action security policy, which can consume as many as four Intel Xeon class CPU cores while achieving only 50 percent of the data rate in a 10GbE network. The demand on the CPU is exponentially higher for more sophisticated host-based networking functions, and the problem will become significantly worse as server connections migrate to 25, 40 and 50GbE in the next few years. Figure 3 shows an example of such efficiency degradation in terms of throughput per dollar and throughput per watt (power) for a typical data path processing scenario.

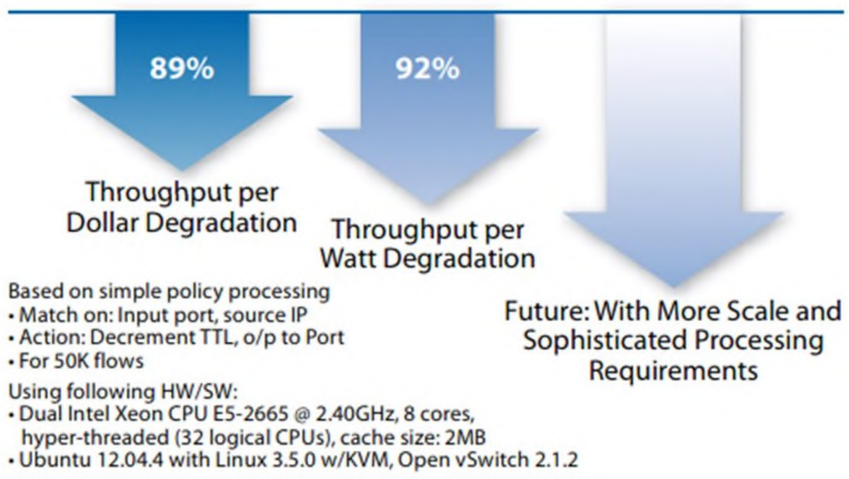

Figure 3: Throughput per dollar and per watt (power) degradation resulting from data path processing in software

There are two additional and significant implications of the proliferation of host-based networking deployments. First, the need for intelligence in the network switching infrastructure for the implementation of sophisticated security and congestion management policies at scale on a per-application and/or per-user basis, is reduced substantially. This is especially helpful since the laws of physics with respect to silicon technology make it difficult to efficiently implement sophisticated policy management in switching silicon, while also increasing bandwidth to multiple terabits-per-second of packet processing and delivering lowest port-to-port latency. This simplification of the switching infrastructure in the data center has facilitated the adoption of so-called “white box” switches with disaggregated networking operating systems, resulting in substantial capital and operational cost savings.

The second implication is how networking appliances, such as load balancers and firewalls, are implemented. With the simplification of the network infrastructure and the desire of data center operators to control the development and deployment of new features, the appeal of COTS server-based applications is growing. These “super servers” being used for the most demanding network functions are sometimes called service nodes or network nodes, particularly by carriers. Of course, many of these same network functions can also be implemented in compute nodes in a distributed fashion utilizing elements of host-based networking such as the kernel network stack and virtual switch data paths.

The Emergence of Hardware-accelerated Host-based Networking

Hardware-based acceleration of networking functions in servers is not new. New classes of server adapters have evolved periodically, driven by increasing data rates and sophistication of networking functions in servers. For example, when the need for stateless offload networking functions became important, so too did the need for a new breed of 1GbE server adapters. Support for simple functions like checksum offload evolved into support for more advanced ones, such as large send offload, receive-side scaling and others, driven by the practical need to conserve dwindling server CPU cycles.

Such application-specific accelerations and offloads have become much more prevalent with the adoption of 10GbE. As a result, application-optimized server adapters witnessed wider adoption. Examples include storage area networking (SAN) optimized adapters utilizing Internet Small Computer System Interface (iSCSI) and Fiber Channel over Ethernet (FCoE), as well as high-performance computing and clustered storage-optimized adapters utilizing kernel bypass and Remote Direct Memory Access (RDMA) offload technologies, such as iWARP and RDMA over Converged Ethernet (RoCE).

More recently, with the rapid emergence of host-based networking at 10, 25, 40 and 50GbE speeds, server adapters called intelligent server adapters or SmartNICs for networking data path acceleration are becoming critical. Such adapters are called intelligent or smart because they bring the best of both worlds – they enable hardware-based acceleration and server efficiencies, but also have the ability to evolve rapidly, a critical new feature to keep the pace of software innovation intact.

The second article in this series will explore the three fundamental technologies being used in intelligent server adapters designed for host-based networking applications, and provides guidance on solutions that deliver the best price/performance.

# # #

Improving Price/Performance with Intelligent Server Adapters (Part 2 of 3)

This is the second article in a three-part series that takes an in-depth look at the promise and pitfalls of using general CPUs in host-based networking applications, and explores the different server networking hardware technologies being utilized to deliver better price/performance for host-based networking. The first article [Add Link] focuses on the potential issues and the importance of hardware-accelerated host-based networking. This article explores the three fundamental technologies being used in these intelligent server adapters, and provides guidance on solutions that deliver best price/performance for host-based networking applications.

Data center operators deploying or evaluating host-based networking applications face the challenge of cost-effectively scaling networks to 10, 25, 40 and 50GbE using COTS-based server platforms. As outlined in our previous article, host-based networking functions such as vSwitches that are implemented purely in software are expensive and inefficient at 10GbE and higher speeds, due to the sheer number of costly and power-hungry x86 CPU cores consumed by data plane processing. In this article, a comparison of host-based networking approaches in the context of price/performance will be discussed.

The underlying problem related to pure software implementations is the fundamental mismatch of the capabilities of the x86 architecture when it comes to packet, tunnels, and flow processing—the basic ingredients of host-based networking. This is because the x86, and other general purpose processing architectures that target compute servers, are by design optimized for running complex and relatively long-running applications in an operating system environment. This is great for server applications, but not necessarily the best strategy when it comes to implementing host-based networking. To understand these priorities for x86 servers, consider a chip level view of the Sandy Bridge device shown below:

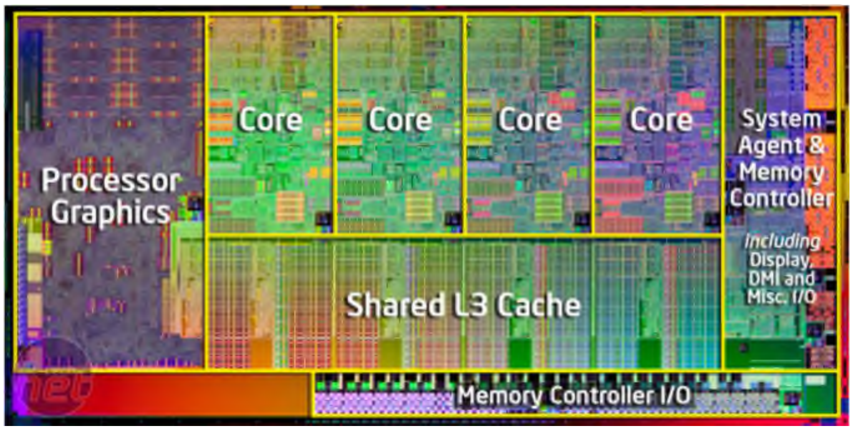

Figure 1: Intel x86: Optimized for servers, not for host-based networking

The need to optimize for general compute applications running in a traditional O/S environment drives a tremendous amount of complexity into the CPU architecture. The x86 cores are loaded with features such as very long and complex superscalar processing pipelines with speculative execution and branch prediction, large caches, and MMUs to support virtual memory, all resulting in a large die area per core. Massive L3 cache is needed to support very large programs and data sets in external memory, but host-based networking data path programs and data are relatively small and would not need L3 in a standalone configuration. Likewise, graphics processing and floating point units in the x86 are not needed for data path processing. Since these server class features dramatically drive up the size of the CPU chip while not increasing work output for host-base networking functions, they strongly reduce the price/performance ratio of the overall solution.

Indeed, applying the x86 to host-based networking is like trying to haul firewood with a Ferrari: it is not the right tool for the job, and far too expensive to boot. This issue has now been widely recognized, and the natural reaction has been to pursue software optimization. Technologies like the Data Plane Development Kit (DPDK) are intended to improve x86 CPU performance in networking applications by improving cache utilization and eliminating interrupt processing overhead. While offering modest improvement by reducing the percentage of processor idles, this approach cannot overcome the fundamental architectural limitations described above, so further improvement will be effectively quite limited.

Alternatives to x86 Software-based Implementations

The failure of host-based networking software applications on x86 CPUs to deliver desirable price/performance has motivated the pursuit of alternative solutions capable of scaling cost-effectively to 10, 25, 40, 50, and 100GbE line rates.

One such alternative is the intelligent server adapter equipped with a MIPS or ARM based multicore system-on-chip (SoC). In this model, the SoC implements the host-based networking data path, entirely in software running on the SoC processing cores. While this offloads the server, it does not change the fundamental processing paradigm: The SoC approach suffers from the same fundamental architectural limitations as the x86. This is because the architecture of these SoC devices is first optimized for the server market, and then re-purposed to server adapters, so the same issues listed above for the x86 inefficiency apply here as well. Effectively, this approach is just a re-distribution of processing resources, while doing very little to improve the efficiency and price/performance of the overall solution. This is evident in the fact that SoC based server adapters struggle to achieve line rate at all but the largest packet sizes in current implementations at 20Gbps.

Another alternative that is often discussed is also a familiar one: the use of Field Programmable Gate Arrays (FPGAs). Recent papers have described the benefits of FPGAs for accelerating specific algorithms related to web search in the data center[1]. There have even been suggestions to apply the same approach to data plane processing for host-based networking. But efficient and cost effective use of FPGAs for host-based networking is yet to be proven and seems unlikely to receive widespread adoption for various reasons noted below.

FPGAs are suited for well-defined tasks that are repetitive and fine-grained in nature such as image and signal processing, compression/decompression, and cryptography, to name a few. They can often perform these tasks more efficiently than software running on general-purpose processors. However, FPGAs are not well-suited for the complex, variable, and irregular processing tasks that are the hallmark of packet processing. Required functionality such as branching, bit manipulation, encapsulation, and filtering are just a few features that cause great difficulty for FPGAs when trying to implement network data paths.

In addition, there is a huge area efficiency penalty for FPGAs when compared to standard ASIC technology that is difficult to overcome outside of very specific use cases. It is a fact the programmable interconnect infrastructure in FPGAs burns up a large amount of area of the die, leading to about 20-30x less effective logic gates per unit area and 12x more dynamic power per equivalent function, when compared to ASIC based designs[2]. Given that the upper limit of die area for server adapters is driven by the common denominators of cost and power consumption, this means that the FPGA is at a significant disadvantage when it comes to performance efficiency.

In addition, one of the main purported benefits of FPGAs, the ability to adapt the functionality via reprogramming, is often quite limited in practice. Significant changes made to the data path may not fit or route in the target device, or hit the same target frequency of operation. Furthermore, FPGAs are typically programmed using esoteric hardware description languages such as Verilog or VHDL and require hand coding for good performance. Efforts to improve this by support for programming of FPGAs with C via OpenCL and other approaches does promise to simplify development, but at the expense of further efficiency, and eroding price/performance even further.

One needs only to look at past history to observe that FPGAs have always been relegated to niche applications when it comes to networking data paths. Often, they are used as a stopgap until more efficient purpose-built solutions are available. In fact, FPGA use is often a leading indicator that there is a product gap somewhere, and if the gap is in an area with sufficient market size, purpose-built solutions will inevitably be developed.

For all of the above reasons, it is clear that intelligent server adapters equipped with multicore SoCs or FPGAs do not offer the scalability and extensibility required to accommodate the host-based networking applications of today and the future. This is, of course, a familiar theme. Time and again, while the industry has tried to reuse existing technologies for new applications, it has proven necessary to accommodate new network acceleration and efficiency-of-scale requirements with new and purpose-built technologies.

Purpose-Built Solutions Evolving to Become Mainstream

When available solutions are unable to meet the demands of emerging and compelling use cases, purpose-built solutions inevitably evolve to supplement and sometimes supplant their use. IP routers were implemented all in software on general purpose CPUs in the 1990s, but the explosion in traffic growth driven by the internet led to the need for higher performance and scale, and the network processor was born. ATM evolved as a purpose-built and targeted technology intended, in part, to accommodate the desire to converge multiple traffic types. MPLS evolved next as an extension to Ethernet that incorporated the best of ATM as a superior solution for scaling Layer 2-and Layer 3-based VPNs. Initial implementations often these technologies often occurred in FPGAs but very quickly ASSPs were developed that could perform these functions with better price/performance which led to mainstream adoption.

A similar evolution occurred with InfiniBand and RoCE. RoCE adapters were purpose-built for low latency and large-scale data transfers with low CPU utilization. Because the solution delivered superior price/performance and scalability, it was able to overcome what had been perceived as a significant hurdle: Use of the InfiniBand transport layer and IBTA-defined verbs vs. the far more familiar TCP/IP and traditional sockets interface. Its advantages prevailed and adoption gradually grew, and RoCE has now been enhanced in version 2 to add support for routing and deployment across Layer 3 networks. While RoCE was initially implemented primarily in software on servers, the processing burden was very high, which drove specific solutions in the form of server adapter ASSPs that supported RoCE offload directly in hardware, now mainstream.

The evolution of purpose-built technologies from specialized to mainstream deployments as a more cost-effective means to accommodate changing needs is depicted in Figure 2. In addition to IP/ATM/MPLS and RoCE, the figure also shows the evolution of purpose-built 3D graphics technologies into mainstream GPU-based products that are now pervasive in PCs, providing another great example of a function that was initially implemented in software on servers, then moved to a purpose-built accelerators, and finally to mainstream adoption in the form of the GPU adapter. The same evolutionary process is also beginning to occur for host-based networking use cases with the advent a new purpose-built technology: the Network Flow Processor, or NFP.

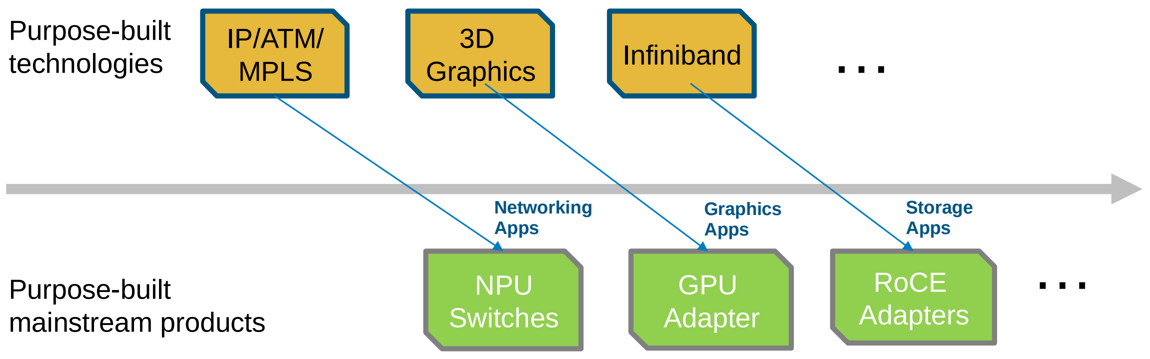

Figure 2: Purpose-built technologies find their way to mainstream deployments

Network Flow Processors: Purpose-built for Host-based Networking

Overcoming the performance and scalability limitations of multicore SoCs and FPGAs requires addressing the root cause of these limitations. For this purpose, Netronome has developed the Network Flow Processor, or NFP.

Intelligent server adapters based on NFPs can efficiently scale from 10Gbps to 100 Gbps of throughput, delivering more than an order-of-magnitude performance improvement over existing software-based solutions. Figure 3 below shows a throughput comparison for a common host-based networking data plane application, Open vSwitch (OVS). As shown, the NFP-based intelligent server adapter delivers more than a 20-times improvement in packet throughput for the same amount of x86 CPU resources (single x86 core), thus dramatically improving the price/performance equation.

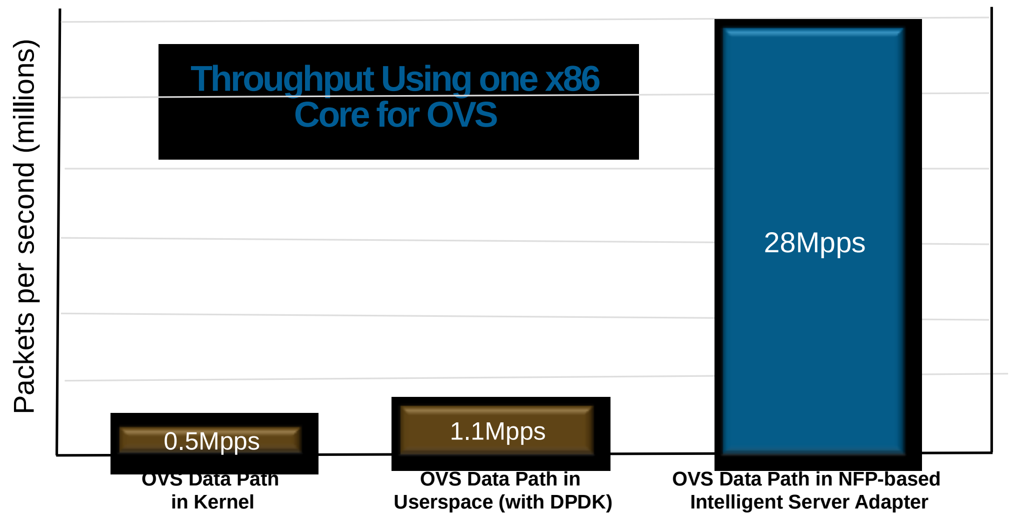

Figure 3: Host-based networking performance using NFP-based Intelligent Server Adapter

While we expect other intelligent server adapters based on MSOC or FPGA approaches to deliver at least some improvement in price/performance, not all such adapters are created the equal. The purpose of the third and final article in this series, therefore, is to outline several important characteristics that are helpful when evaluating an intelligent server adapter for use in host-based networking applications.

[1]"**A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services**", Microsoft

[2]**"Measuring the Gap between FPGAs and ASICs",** Ian Kuon and Jonathan Rose, Department of Electrical and Computer Engineering, University of Toronto

# # #

Intelligent Server Adapters: Key Ingredients for Success (Part 3 of 3)
This is the third article in a three-part series that takes an in-depth look at the promise and pitfalls of using general CPUs in host-based networking applications, and explores the different server networking hardware technologies being utilized to deliver better price/performance for host-based networking. The first article [Add Link] discussed host-based networking and the potential issues when using general purpose CPUs to implement the data path, and discussed the importance of intelligent server adapters for hardware-acceleration and server offload. The second article [Add Link] explored the three fundamental technologies being proposed for these intelligent server adapters, and provides guidance on the solutions that deliver the best price/performance for host-based networking applications. This final article examines the attributes and key ingredients for success that will enable the intelligent server adapter to become a mainstream technology in data centers for years to come.

Many instances of data plane processing have evolved in recent years, both in the open-source community and in commercial deployments on the premises of data center operators. For example, the Open Virtual Switch or OVS[1] user space and kernel components have evolved to introduce more sophisticated and scalable tunneling and flow processing capabilities. The OVS kernel modules handle tunneling and switching. The kernel also implements a fast caching mechanism for non-overlapping or exact-match flows. More recently, support for tunnels such as VXLAN[2] and wild carding match-action rules have been added to the data path. In commercial deployments such as Microsoft Azure, the virtual switch supports network virtualization for tenants as well as sophisticated match-action processing for functions such as load balancing and security. In the Google Andromeda network, packet processing nodes provide match-action flow and data plane processing for firewall, security, rate limiting, routing and other functions.

Data plane processing for host-based networking has evolved in the host or server domain, implemented in software on general purpose CPUs such as the x86 instruction architecture. However, unlike traditional processing tasks executed on such general purpose CPUs, data plane processing and more specifically tunnels encapsulation and decapsulation, and match-action based flow processing are unique in nature. This results in significant inefficiencies when they are executed on general purpose CPUs. Alternative solutions have been proposed, such as running the data plane in a multi-core SoC or FPGA, but each of these approaches provides only marginal gains, as discussed in our previous article.

Implementing data plane processing with efficiency and scale requires processing cores and product architectural elements that are purpose-built, with special ingredients. Next we discuss seven such key ingredients for success.

Key Ingredient # 1: Processor Multi-threading

Flow processing requires access to memory such as DDR3 or DDR4 based RAM. To assist processing in CPU cores, hardware-based accelerators are used for repetitive or specialized functions such as cryptography or hashing. With single threaded processing, as with general purpose CPUs (like standard x86, MIPS and ARM cores), memory and accelerator access latencies result in wasted CPU cycles. For example, accessing DDR3 memory could take hundreds of CPU cycles, and access to hardware accelerators can take even longer, leaving the CPU core idle and effectively useless for this extended period of time. This will often reduce the effective CPU utilization down to 10-20% for typical host-based data plane processing tasks. Software custom coding techniques can be used to fill latency gaps, but these changes are time consuming and cumbersome, and they negatively impact the portability of the software.

An ideal solution to the problem is to implement processing cores that are highly multi-threaded. When processing cores are multi-threaded (for example eight threads per core), the processor pipeline can always be executing useful instructions instead stalls or idles. As a result, multi-threaded processing gain can be up to 800% over single threaded machines when

[1] Open Virtual Switch is an open source community based development. See www.openvswitch.org

[2] VXLAN is a network virtualization related specification being developed in the IETF and stands for Virtual Extensible Local Area Networking

memory or hardware accelerator access requirements are significant, as in the case with typical data plane processing requirements in host-based networking and emerging NFV applications.

Key Ingredient # 2: Many Processor Cores Better Than a Few Faster Cores

General purpose CPUs are typically optimized for highest processor clock speeds at the expense of power and area. For example, big and complex pipelines with more than fifteen stages, out-of-order execution, and branch prediction capabilities are common in such CPUs. Large caches are also needed because of the lack of multi-threading, to reduce the effect of memory latency, as explained earlier. When such general purpose CPU cores are packed into a single silicon die, as in MIPS or ARM-based multi-core SoC implementations, the effective performance gain is not as high as packing a significantly larger number of smaller processing cores in the same silicon die. In other words, more of optimized multi-threaded processing cores in a silicon is better for data plane processing than fewer higher performance general purpose CPU cores with no or lower number of threads and large caches. The use of large processor cores becomes a significant overhead when price and power constraints are placed on server adapter designs, as seen with general purpose servers used in compute nodes.

Key Ingredient # 3: Memory and Accelerator Multi-threading

Efficient access to memory and hardware accelerators is critical in data-intensive flow processing, and the challenge is only exacerbated with larger numbers of flows and complex processing (such as the number of tuples used in matches and complexity of actions). Such requirements are bound to become pervasive in data centers with the growing need to support more users, tenants and applications, and the requirements for stringent policies related to security and service levels. While faster access to memory is important, multi-threaded access to memory is even more important. A multi-threaded memory subsystem with hardware accelerators can ensure that the processing cores do not stall. An example of such an efficient design is where multiple banks of SRAM are used with multiple crossbar inputs with high bandwidth. Access to such SRAM banks are further accelerated using high performance tightly coupled dedicated hardware engines that perform critical functions such as atomics, statistics, lookups, and load balancing.

Key Ingredient # 4: High Performance Distributed Mesh Fabric

The multi-threaded processing cores, hardware accelerators and multi-bank memory units described above need to be well-synchronized and provide high performance while avoiding stalls in processing. Traditional shared bus architectures suffer from bandwidth saturation and contention issues under load when accessing shared resources. This can be avoided by using an efficient high performance distributed mesh fabric that with multi-terabit bi-sectional bandwidth between processing elements. Such a distributed mesh fabric will avoid contention and bus saturation issues that are common with shared bus architectures seen with general purpose CPU-based SoC solutions.

Key Ingredient # 5: Optimized Programming Tools for Host-based Networking

While at first glance general purpose CPU cores seem easy to program, for example by using standard C-based programming tools, the difficulty and complexity increases substantially when attempting to parallelize applications and scale performance. So in this sense, they do not have good support for development of optimized data plane processing applications. When programming multi-threaded processing cores, it is critical that robust, easy-to-use, C-based programming tools that support parallel programming environments are available that provide thread-level visibility during programming, and allow creation of data plane processing programs that are optimized for multi-threaded operation.

In addition to C-based programming tools, it is now becoming possible to support high level programming languages such as P4[3] that make the description and coding of data path functions much simpler and less time consuming. Using the open P4 language, designers can write concise programs that can flexibly define match and action processing to quickly implement new protocols such as emerging network overlays. P4 is also hardware agnostic, so it can be re-targeted to different technologies and implementations, providing they support the P4 environment.

Key Ingredient # 6: Hitting Compute Node Economics

Intelligent server adapters are evolving in a natural way, starting with niche, lower-volume applications, with the promise to grow into high volume, mainstream deployments. Initial implementations using multi-core SoC silicon have found their way into appliances and purpose-built servers that are sometimes called service nodes or network nodes. There are also instances with the use of NPUs and FPGAs in such applications. Because the volume of deployment for service nodes has not been very large, data center operators have been willing to pay premium pricing for programmable server adapters capable of data plane processing.

However, with host-based software-defined networking (SDN) and network functions virtualization (NFV) technologies moving toward mainstream adoption, the need for intelligent server adapters in the much higher volume compute nodes is expected to rise significantly. This will require intelligent server adapters that will exhibit much better characteristics when it comes to price/performance as compared to the early service node implementations. Specifically, such adapters will have to operate at wire-speed and within the PCI express power envelope of 25 watts in most servers deployed today. And most importantly, they will have to be reasonably priced to support the volume economics of compute node servers. This means that silicon technology and data plane processing architecture used in programmable server adapters must allow for performance, scale and economics. Key ingredients 1 through 5 highlighted above are required to enable needed data plane processing requirements at 25, 40 and 50GbE bandwidths while hitting compute node economics expected by data center operators.

Key Ingredient # 7: Ready Software Ecosystem for Mainstream Adoption

In addition to meeting performance, feature, price and power requirements, mainstream adoption of programmable server adapters will require a well-supported software ecosystem. Specifically, the server operating system kernel, user space and virtual switch networking software stacks must support installation and operation of such server adapters that can

[3] P4 is an open source higher level programming language that is hardware agnostic. See www.P4.org.

offload data plane processing such as virtual network tunneling and match-action related flow processing.
Ease of integration with existing server applications, open source software leverage, and maximal feature velocity are of paramount importance for a successful solution. As an example, offload architectures that work with existing open source solutions such as Open vSwitch, as opposed to replacing them with proprietary or forked solutions, will undoubtedly prevail. A critical point is that any new features, when implemented in the open source community, must be up-streamed into the appropriate open source mainline trees (such as in www.kernel.org or www.openvswitch.org). Commercial operating systems and hypervisor distributions from well-known operating system vendors should also include datapath offload capabilities such as the above. Operating system vendors must support qualification of programmable server adapter vendor-supplied device drivers and associated software with their distributions to enable seamless operation of such adapters and their mainstream adoption by data center operators.

Mainstream Market-Ready Intelligent Server Adapters

Host-based networking deployments are expected to drive mainstream adoption of intelligent server adapters. Such adapters have to be purpose-built and support key ingredients for success. We discussed seven such key ingredients for success that span architecture and technology in such server adapters as well as volume economics and software ecosystem requirements. Netronome's innovative C and P4 programmable flow processors that utilize a unique small-core highly multi-threaded architecture are built to specifically solve the scaling and efficiency challenges of host-based networking. They are unique in that they possess the key ingredients for success and are in lock step with important developments occurring in the open source as well as commercial operating system and hypervisor ecosystems. Stay tuned for more details on Netronome's flow processor architecture and intelligent server adapters in an upcoming article.

# # #