

Optimized Networking with Multicore

Table of Contents

Executive Summary	1
The High-Performance Challenge	1
Multicore Performance	2
Multicore Processing Methods.....	2
Network Protocol Processing with Multicore.....	3
Multicore Design Considerations	4
Caching and Hashing.....	4
Multicore Performance Benefits.....	4
Conclusion.....	5
Notes	5

Executive Summary

Performance and cost have always been critical metrics for evaluating networking equipment. But performance has multiple dimensions, including throughput, latency, and CPU utilization. Even in systems with less than 1GB of capacity, predictable response time and available CPU cycles to run applications are important. Multicore silicon offers an opportunity to optimize both performance and cost. By efficiently distributing networking functions across multiple cores, systems can realize greater throughput, lower CPU utilization, smaller footprint, and lower costs than previous generations. This paper illustrates how the networking industry will be revolutionized through the efficient integration of operating systems, network stacks, and multicore silicon.

The High-Performance Challenge

Moore's Law implies that processor power will double about every two years¹. This has broad implications to any equipment using microprocessors, including networking equipment. More powerful end nodes can process data more quickly, driven by the ever increasing demand for networking bandwidth. We have seen local area network (LAN) speeds increase a thousand times over the past 15 years.

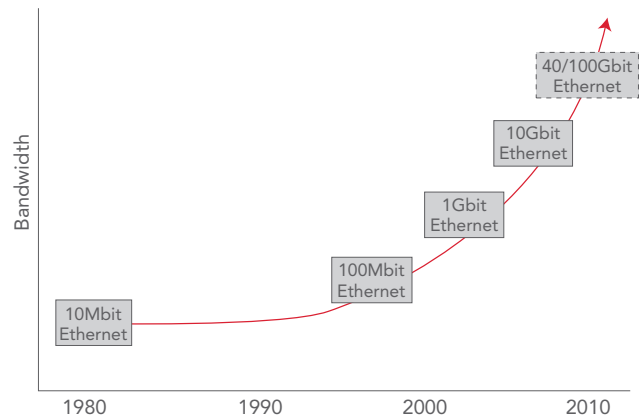


Figure 1: Evolution of LAN data rates

Wide area network data rates still lag behind LAN speeds but have also increased exponentially.

The most rapidly developing networking technology is wireless, which offers the convenience of connectivity without requiring users to be tethered to a wire or fiber.

One could argue that the demand for multimedia content has been around forever, simply waiting for networking technology to catch up. Video and audio files are not only much larger than plain text but they are sensitive to delay.

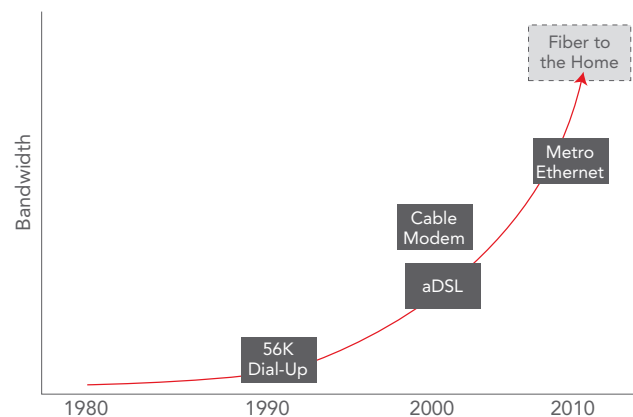


Figure 2: Evolution of WAN data rates

The merging of voice, video, and data require more sophisticated internetworking devices with low latency requirements. Home gateways now need to process a mix of Internet access, Voice over Internet Protocol, and streaming video.

Likewise, handheld devices such as Apple's iPhone blend voice, data, music, Internet access, and video in an even smaller form factor.

All of these trends create a common demand for high bandwidth, low-latency networks, not only for end-user devices but also for the edge, aggregation, or core elements. Equipment manufacturers are faced with the challenge of providing high-performance platforms in a market where margins are squeezed and development cycles shortened. A new approach is required to meet these challenges. Multicore networking offers the solution.

Multicore Performance

Over the past several decades it has been proven that processor power doubles every two years. However, the processor speed curve is beginning to flatten as in recent years processors have not been able to make use of the increase in available transistors due to heat and power limitations.

But multicore processing offers a new paradigm. By using multiple cores in parallel, processing power can be increased to meet high-performance demands.

Most leading processor silicon producers have introduced multicore silicon, where multiple processing cores are

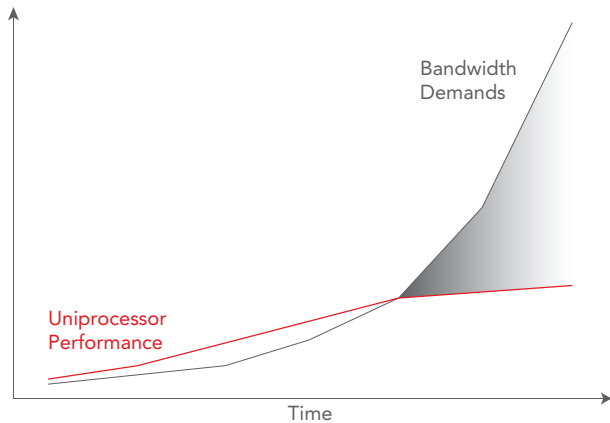


Figure 3: Single processor performance gap

integrated into a single chip. Virtual cores, or threads, can provide further subdivision of core resources by facilitating very fast context switching between tasks.

The power of multiprocessor chips ranges in clock speed and number of cores. Chips with 16–32 cores are within reach today. Many of these chips have integrated network processing functions that reduce latencies traditionally introduced by network protocol software.

Multicore Processing Methods

Multicore software can be implemented in various forms². In a system implementing symmetrical multiprocessing (SMP), multiple cores are essentially interchangeable as they execute the operating system and tasks. A variation on SMP uses affinity or CPU reservation to bind specific tasks and specific cores, which effectively makes them dedicated processors.

Asymmetrical multiprocessing (AMP) is typically used to describe systems in which multiple operating systems are implemented. Supervised AMP uses virtualization³ to abstract processing elements such as memory, cores, or devices.

Software must be designed to take advantage of the new silicon. A common misconception is that applications written for single processor environments will automatically perform better in multiprocessor environments. Consider an example in which a robot arm is used in an assembly line to move boxes onto a pallet. When powered by a single processor, the arm can place 12 boxes per minute on the pallet. If the same system is powered by a multicore processor in SMP mode, the robot arm does not move any faster; it still places 12 boxes per minute on the pallet. However, if software is written for multicore processors, the system can use the extra processing power to perform other tasks. For example, the second processor could control a second arm that could be interleaved with the first arm, doubling effective productivity to 24 boxes per minute. The second processor may also be used to control a conveyor belt, process telemetry feedback, or offload tasks from the first processor that allows it to increase productivity to 15 boxes per minute. The salient point of this example is that the transition to multicore processing alone does not automatically increase performance; attention must be given to the interaction of the cores to harness the power of multiple processors.

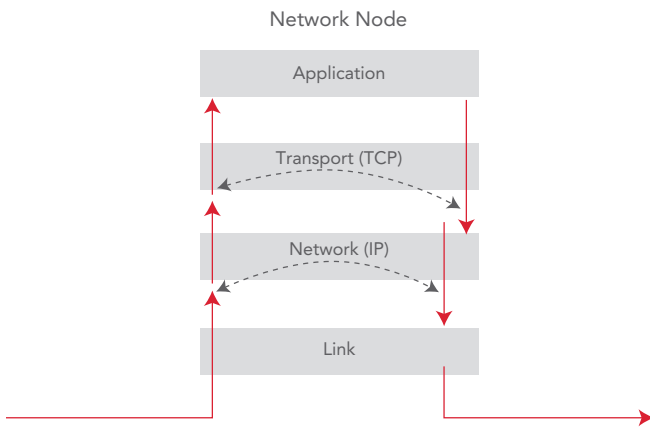


Figure 4: Traditional monolithic packet processing

Network Protocol Processing with Multicore

The most common networking protocol stacks use the Internet Protocol (IP) and Transmission Control Protocol (TCP). These protocols are widely used in the Internet and in virtually every industry using network connectivity. Likewise, networking equipment must support a common set of protocols to support the infrastructure that keeps these industries connected. Traditionally these protocols are treated as a monolithic stack that processes packets uniformly.

The steps required to process an incoming packet can be thought of as a state machine. First a packet is received on the link layer (usually Ethernet) interface. Then it is queued for the network or IP layer. The IP layer must then determine whether the packet is destined for this network element or should be forwarded on. This layer may also be required to perform cryptography as part of IPsec or Internet Key Exchange (IKE) processing. If the packet is destined for this network element, it is passed up the stack to the next layer, which is usually the TCP or the User Datagram Protocol (UDP). Additional security functionality may be required here as part of Secure Sockets Layer, or SSL processing. If the encapsulated data is still intended for this node, it is passed on to an application

protocol such as FTP, SMTP, Telnet, or HTTP. In a single processor system or even a pure SMP system, all of these network processing states contend for processor cycles.

A single distributed network stack architecture will not be ideal for all scenarios. For example, the repetitive steps used for packet forwarding are different than the steps used for packet termination. Consider the difference between a layer 3 switch or gateway, which forwards packets between interfaces, and a web server, which receives (and terminates) requests and returns pages of HTML data. Optimizing the stack may require different architectures for each of these scenarios.

Another design choice must be made regarding which layer in the protocol stack is to be distributed. If most connections will be managed at layer 4 (TCP), distributing multiple instances of TCP across processor cores may be advantageous. However, multiple instances of more complex protocols usually require more interaction between cores, which can lead to added latency and memory bandwidth constraints. These may not be significant for bandwidth of 1 to 2 gigabits but can become a limiting factor for scaling the system for greater throughput. The cost/benefit analysis of these considerations can only be made in the context of the overall system goals.

In high-performance networking equipment such as switches, routers, and gateways, much of the network traffic is forwarded at the IP layer. As a highly repetitive task, packet forwarding is an excellent candidate for multicore processing. The first step is to decouple the forwarding code of the IP protocol from the address establishment logic. When a packet address is seen for the first time, it is recorded in a table. Because this only happens once, table setup can be done using the “slow path” by a multipurpose core that functions as a traditional network stack processor in the operating system. The fast path processor (or forwarding core) simply looks at every incoming packet’s destination address and checks for a match in its cache table. If found, the fast path processor quickly determines the appropriate output port and queues the packet appropriately.

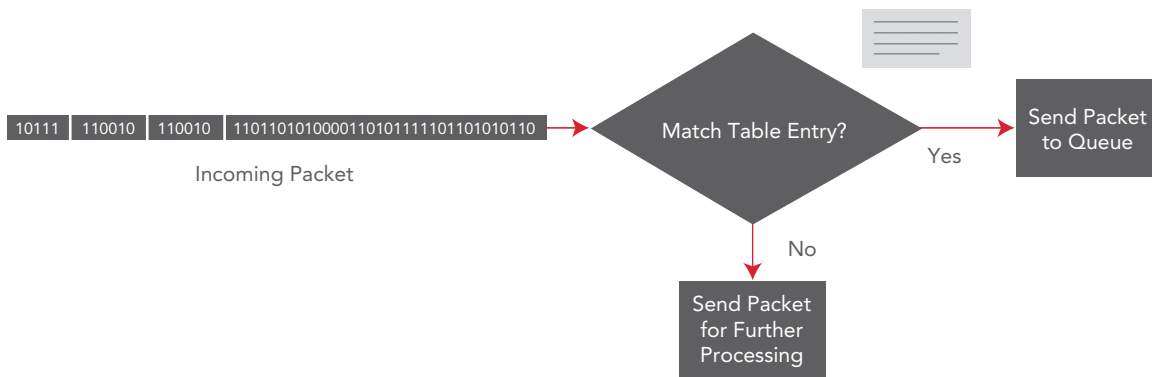


Figure 5: Logic states for packet processing

Classifying IP packets in this way lends itself to efficient layer 3 network functions including forwarding, Network Address Translation (NAT), access control lists, IPsec, and other cryptographic uses. Some silicon may have specialized engines for cryptography, which can execute in parallel with a core CPU.

Although the code required to perform this fast path forwarding is relatively simple, high traffic rates from multiple gigabit interfaces can still overwhelm the processing core. Adding more forwarding cores may help alleviate some of the throughput problem, but unless some other design techniques are adopted, system performance will reach a limit imposed by other bottlenecks.

Multicore Design Considerations

Designing the network stack to be distributed across multiple cores can dramatically increase system performance in networking equipment, but only to a point. When cores share a common memory space, only one can be updating the information at a time. The other must wait to gain access. These interlocking safeguards are often implemented in software using semaphores, spinlocks, and mutual exclusion. When multiple processors are forced to wait on each other to modify data structures in shared memory, adding more cores exacerbates the problem by forcing more collisions. This is often the case when performance curves show a “knee,” or a flattening out after a certain point in the throughput.

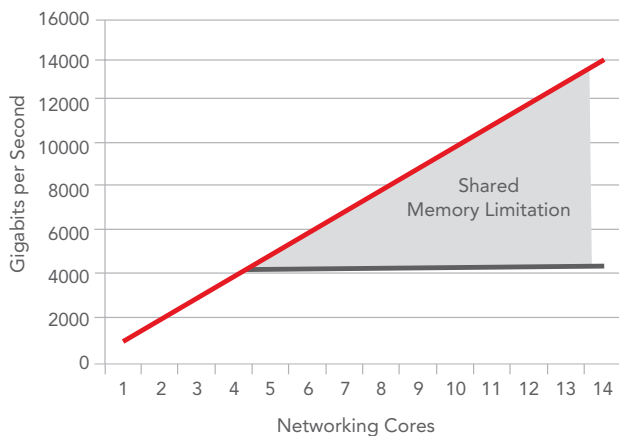


Figure 6: Shared memory bandwidth limitations

Careful consideration must be given to architect code to minimize locking. Wherever possible, silicon features that enable lockless code should be used to avoid software performance limitations.

Caching and Hashing

For decades, instruction caching has been studied and refined to enable more efficient processing. The highly repetitive tasks that are good candidates for multicore offloading are also susceptible to subpar performance if instruction caching is not

optimized. Consider an IP forwarding core that can process 1.4 million packets per second. A single cache miss can cause this loop to require 15% more clock cycles, which can lower the maximum processing by an equivalent amount. Tailoring code to match instruction caches can be meticulous work, but using caching efficiently can make a significant difference in the performance of protocol processing.

Likewise, data tables such as port mappings, address translations, flow information, and security associations must be done efficiently in order to avoid a knee in the performance curve. Hashing can be an efficient way to access such information but only if sufficient space is allocated to avoid collisions. If two entries reduce to the same hash value, access can become inefficient. This is inevitable unless the system is designed and configured in advance to accommodate the anticipated traffic rates.

Multicore Performance Benefits

The benefits of tailoring a network stack to use multicore technology are twofold. The first and most obvious metric is throughput. Throughput is typically measured in either packets per second or megabits per second. A common misconception is that Gigabit Ethernet can support 1 gigabit of data per second. Some of the bandwidth is taken up by interpacket gaps and header information.

Preamble: 8 bytes
 Interpacket gap: 12 bytes
 20 bytes

On a 1Gb-per-second link with 64 byte Ethernet frames, 20 + 64 = 84 bytes are actually sent on the wire. In other words, 20/84 = 23.8% of the capacity of the line (238Mbps) is used up by overhead, leaving 76.2% (762Mbps) for transmitting data. When frame sizes are larger, they are sent less frequently and the overhead represents a smaller percentage of the available bandwidth. Although these overhead bits could be included in measurements, convention is to measure only payload data and the cyclic redundancy check (CRC). The following table shows the maximum frame rate and amount of useable data for each frame size.

Frame Size	Throughput	FPS
64	762Mbps	1,488,000
128	865Mbps	844,595
256	928Mbps	452,900
512	962Mbps	234,962
1024	981Mbps	119,732
1280	985Mbps	96,154
1518	987Mbps	81,274

Table 1: Theoretical maximum payload for 1Gb Ethernet in megabits (Mbps) and frames per second (FPS)

Common benchmarks include packet forwarding, in which packets are received in one interface and forwarded on to another; and packet termination, in which packets are processed and terminated. Depending on the type of network equipment being tested, one test may be more helpful than another for measuring performance.

Figure 7 shows some IPv4 packet forwarding rates measured on a 500MHz Cavium OCTEON 3860 running Wind River's VxWorks 6.7 platform.

Frame Size	Traditional IP Forwarding	Multicore Network Acceleration	Percent Increase
64	18	762	4133%
128	34	865	2444%
256	64	928	1350%
512	127	962	657%
1024	277	981	254%
1280	343	985	187%
1518	410	987	141%

Measured with VxWorks 6.7 EAR release on Cavium 3860, 500MHz, two cores/one forwarder; VxWorks core is 99% to 100% idle during test

Figure 7: Network packet processing with multicore acceleration

In addition to network throughput, the performance of the overall system is dependent on the amount of processing power available for other system tasks. If all of the system's resources are consumed with packet processing, even simple administrative tasks can either starve or cause performance to suffer. The bottlenecks mentioned previously can become very visible in these tests, even if the desired throughput rate is achieved. For network offload to be truly meaningful, the offloaded tasks must have minimal dependencies and interactions on the core operating system.

Conclusion

Multicore silicon offers a new paradigm for fast packet processing. By combining multiple processing cores on a single chip, networking equipment can be designed with a smaller footprint, lower power consumption, lower costs, and higher performance. However, scalable wire-speed performance can only be achieved if careful consideration is given to the hardware and software architecture. Hardware acceleration features can save precious processing cycles and should be used wherever possible. Making efficient use of cache lines and hash tables is critical to maximizing fast path performance. Symmetric multiprocessing can be efficient but only if care is taken to make networking tasks execute in a highly parallel fashion. Offloading tasks to specific cores can be highly efficient, reaching wire-speed performance and linear scalability.

Notes

1. Moore's Law claims that the number of transistors that can be placed on an integrated circuit will double about every two years. This has been applied to many technology areas along dimensions of performance, capacity, and cost.
2. *Device Software Optimization for Concurrent and Consecutive Systems*, Wind River, <http://windriver.com/whitepapers/>.
3. *Achieving Business Goals with Wind River's Multicore Solution*, Wind River, <http://windriver.com/whitepapers/>.