



White Paper

Packet Processing with
Intel® Multi-Core Processors

Packet Processing with Intel® Multi-Core Processors

General Purpose Processor Exceeds Network Processor Performance

“In all likelihood, multiple processing cores in the Quad-Core Intel® Xeon® processors will enable truly significant improvement in performance density for this type of intensive signaling plane processing.”

Jorgen Trank,
Product Manager
Signaling Solutions,
TietoEnator

Executive Summary

In fewer than ten years from now, industry experts are forecasting network capacity demands will grow by a factor of one thousand¹. Preparing for this data and bandwidth boom, service providers are looking for cost-effective equipment that is more flexible and scalable. They are turning to open standards-based IP networking equipment that increases the interoperability and reuse of hardware and software components and ultimately reduces deployment costs. Open standards are the foundation for the next-generation wireless networks, called 3GPP Long Term Evolution (LTE), which will increase IP traffic throughput and decrease the number of network elements by consolidating functionality.

Supporting a standard-based approach, embedded Intel® architecture components have a long history of serving wired, wireless and networking infrastructure. They perform a broad range of functions including *control plane* (radio network controller), *application server* (home location register) and *security* (firewalls and VPN). But with the latest Intel® Multi-Core Processors, equipment manufacturers are dedicating some of the extra CPU cores to data plane and packet processing and achieving throughputs over 1,260 megabits per second (Mbps). Traditionally, this level of data plane performance necessitated the use of dedicated hardware like network processors, ASICs and FPGAs. Today, developers are combining control and data plane functions on a single board and eliminating packet processing boards, which lowers cost and speeds up content processing.

This white paper focuses on consolidating control and data processing on general purpose multi-core processors. It describes methods to optimize packet processing software and enable network elements to attain compelling performance running on embedded Intel architecture-based platforms. Benchmark data is also provided for two prototypes, a radio network controller (RNC) and standard packet processing, running on a single board based with Quad-Core Intel® Xeon® processors.

Table of Contents

Radio Network Controller Prototype Implementation	3
Network Processor Implementation	3
Intel® Architecture Implementation	4
Software Stack on Intel® Architecture	4
RNC Test Results	5
RNC Prototype Software Optimizations	5
Next Generation Mobile Networks	6
Flow Processing Prototype	6
Prototype Test Configuration.....	7
Packet Forwarding Test Results	7
Impact of Control Plane Processing	8
Packet Processing Performance Optimizations.....	9
System Partitioning Approaches	10
Para-Partitioning Overview.....	10
Advantages of Para-Partitioning.....	10
Conclusion	11
Appendix	12
Definition of RNC Terms.....	12
System Configuration.....	12

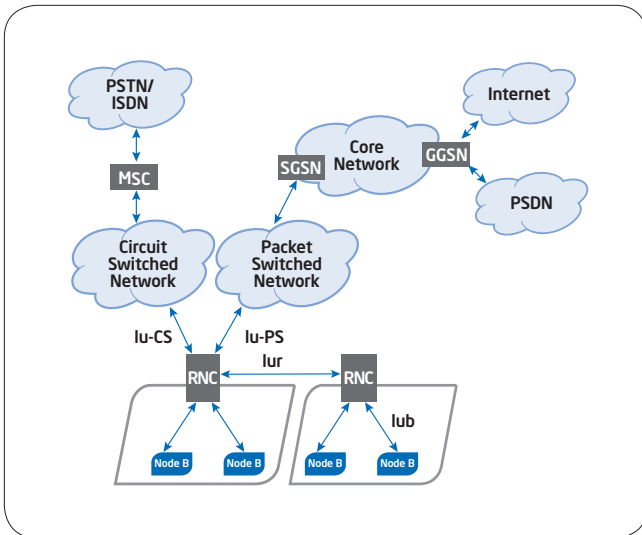


Figure 1. 3G Wireless Network Architecture.

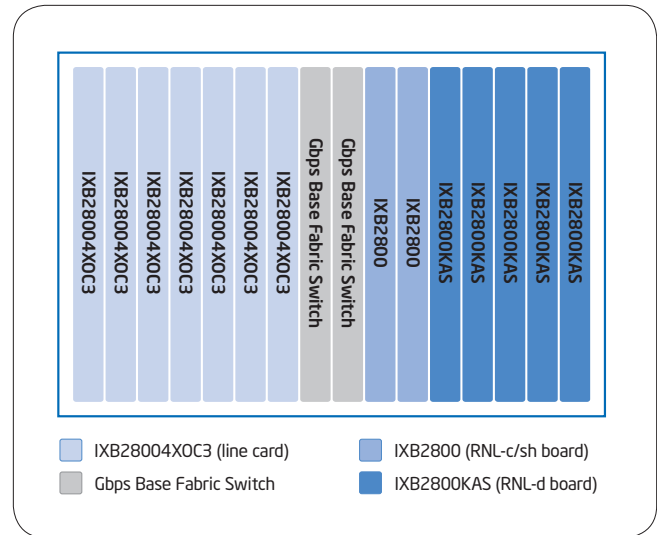


Figure 2. RNC Configuration - Network Processor-based Data Plane.

Radio Network Controller (RNC) Prototype Implementation

This section describes radio access networks, compares RNC performance for systems using network processors and general purpose Intel® Architecture Processors, reviews useful software optimizations and discusses next-generation wireless infrastructure. The RNC benchmarks show that general purpose processors can exceed network processors in packet processing performance.

A 3G Radio Access Network consists of a variety of network elements, as shown in Figure 1. The RNC and Node B connect wireless subscribers to circuit-switched and packet-switched networks. This connection comprises separate paths for call data and signaling, called the data plane and control plane, respectively. In addition, the RNC provides radio resource management and call handover control. Full descriptions of radio access network elements are included in the appendix.

The RNC supports four primary interfaces that carry network traffic to other elements, as shown in Figure 1:

- Iu-CS to the circuit-switched network (uplink).
- Iu-PS to the packet-switched network (uplink).
- Iub to a Node B (downlink).
- Iur to another RNC (downlink).

The following discussion compares two implementations, based on Intel® network processors and multi-core Intel Architecture Processors, modeled to host over 500,000 subscribers. The prototype includes processing for the radio link control (RLC), media access control (MAC) and framing protocols. The benchmark testing exercises the most demanding workloads, the uplinks from circuit-switched and packet-switched networks.

Network Processor Implementation

An RNC implementation, using Intel NetStructure® IXB2800 3G boards, is illustrated in Figure 2. This configuration is based on a traffic model and performance validation results from Intel NetStructure IXB2800 3G board testing performed at Intel. These boards, based on Intel® IXP2800 network processors, are designed to provide the performance and density required by next-generation WCDMA RNC data plane solutions.

- **IXB28004XOC3 (eight line cards)** – OC-3/STM-1 interfaces (1:1 sparing).
- **IXB2800 (two RNL-c/sh boards)** – paging and other data exchanges (1:1 sparing).
- **IXB2800KAS (five RNL-d boards)** – data plane for 504K subscribers (one spare).

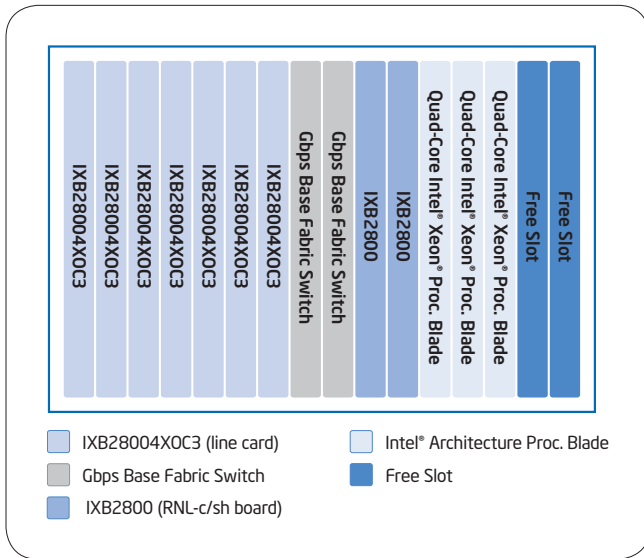


Figure 3. RNC Configuration - Intel® Architecture-based Data Plane.

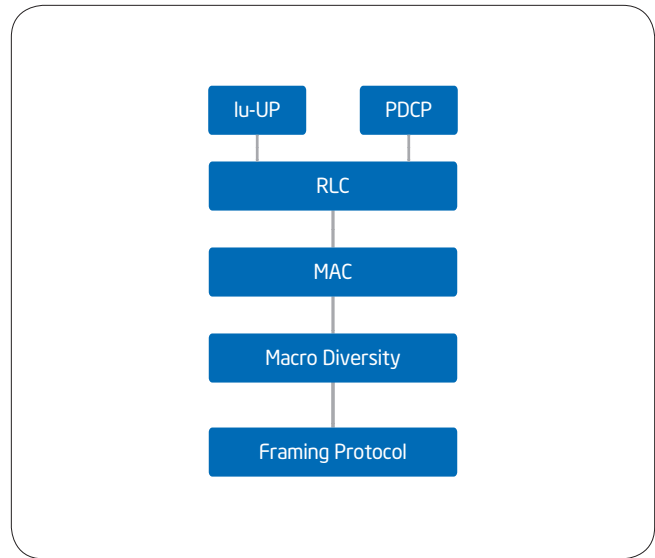


Figure 4. Proof of Concept Software Stack.

Intel® Architecture Implementation

A second RNC implementation uses standard multi-core Intel Architecture Processors for data and control plane functions, as shown in Figure 3. This configuration is the same as the previous implementation, except the five IXB2800KAS boards are replaced with three Quad-Core Intel Xeon processor 5300-based blades (1:1 sparing). As a result, two slots are now free and available for future expansion. Each blade has two quad-core processors – a total of eight CPU cores – running at 2.13 GHz.

Software Stack on Intel® Architecture

The RNC proof of concept, based on Intel architecture, implements a software stack that performs similar functions as the IXB2800KAS RNL-d boards in the network processor implementation, as shown in Figure 4.

The software stack includes the following functionality:

- **User Plane Protocol Layer (lu-UP):** lu-UP provides a reliable transfer mechanism for signaling to and from the core network.
- **Packet Data Convergence Protocol (PDCP):** PDCP is a user plane protocol executed by the core network (CN) line card for header compression and decompression, as well as mapping higher layer protocols to the underlying radio interface protocols.

- **Radio Link Control (RLC):** As part of the Real Time Unit (RTU) software stack that processes user streams, the RLC block provides logical channels for control plane signaling and user plane information. RLC performs error correction, segmentation, reassembly and buffering. This prototype implementation does not support ciphering procedures.
- **Medium Access Control (MAC):** The MAC is the link layer protocol that enables communications between user entities (cell phone users) and the Universal Mobile Telecommunications System (UMTS) terrestrial radio access network and carries both control plane signaling and user data. The MAC performs dynamic scheduling of traffic on shared resources and traffic volume monitoring.
- **Macro-Diversity:** This block manages soft call handover and handles errors arising when several antennas are used for transferring the same signal.
- **Frame Protocol (FP):** A frame protocol is used in the user plane of the Node B interface to define frame structure, messages and procedures for sharing user data and control information.

	RNC Data Plane	
	Network Processor Implementation	Intel® Architecture Implementation
Blades	IXB2800KAS (RNL-d board)	Quad-Core Intel® Xeon® processor 5300 server
Number of Blades		
Active	4	2 (6 of 8 CPU cores active)
Spare	1	1
Total Blades	5	3
Supported Throughput Incoming luCS/PS traffic	1,260 Mbps	1,260 Mbps
Utilization	100%	6 CPU cores 50%
Control plane function	Not supported by IXB2800KAS boards	2 CPU cores available

Table 1. RNC Benchmark Results

RNC Test Results

Both platforms supported the RNC traffic associated with a 504K subscriber network, approximately 1,260 megabits per second (Mbps), as shown in Table 1.^{2,3} The network processor implementation employed four active IXB2800KAS blades, all of which operated at full capacity. The Intel architecture implementation has two active blades, where six CPU cores were responsible for data plane traffic, allowing the remaining two CPU cores to be used for other applications such as control plane processing. The six active cores operated at 50 percent utilization. At slightly higher CPU core utilization, this platform handles even more network traffic, demonstrating that general purpose processors can exceed network processors in packet processing performance.

The Intel architecture data plane implementation requires less the space (three versus five blades), thereby freeing up two blade slots, compared to the network processor implementation.

RNC Prototype Software Optimizations

The following software optimizations substantially improved the performance of the RNC prototype based on Intel architecture.

- **Memory Optimization:** The Linux* kernel supports several memory allocation schemes, some of which provide better performance in packet processing applications than others. For example, the Linux slab cache allocator reduces memory management time by eliminating the need to repeatedly initialize an object for the same purpose. The slab allocator also allows a module to pre-allocate fixed buffers for similarly sized objects,

thus avoiding fragmentation problems. This type of pre-allocation is typical in embedded applications, and in this case it configures the memory that determines the number of subscribers the RNC can support at a time. The Linux slab allocator satisfies these memory requirements with hardware cache alignment, which increases cache utilization and improves performance.

- **Locking Optimization:** Multi-processing systems with processor share resources, like network interface cards, are susceptible to stalls when processors are waiting for locks to be released. This RNC implementation minimizes the amount of time a lock is held, especially in downlink processing. This can be accomplished by allowing modules to share buffers and buffer queues for both uplink and downlink processing, which avoids bottlenecks.
- **Single Buffer Copy:** Another key optimization strategy is to minimize buffer copies or buffer allocation. For example, when the Macro-Diversity controller (MDC) operates on the uplink processing path, it's common to combine buffers from multiple sources and pass them to the lu-UP for transmission to the circuit-switched network; this prototype implementation ensures that there is only one buffer copy performed in the complete processing path. This is achieved by passing on a buffer map along the processing path rather than copying buffers at every stage. In other words, the MDC combination process that merges buffers is executed in a delayed or 'lazy' fashion, and at the latest time possible.
- **Affinity Setting:** Performance improves significantly through the use of affinity settings that tie software processes (e.g., threads) and interrupts to specific CPU cores, which maximizes CPU cache locality and memory efficiency.

Next-Generation Mobile Networks

The next generation Radio Access Network, currently referred to as 3GPP Long Term Evolution (LTE), is the fourth generation mobile standard focused on upgrading the UMTS. This change will help enable the migration, from fixed to mobile networks, of Internet applications such as Voice Over IP (VoIP), video streaming, music downloading, mobile TV and so on.

To meet these objectives, LTE networks must deliver greater capacity and higher performance than 3G networks, which is driving a reduction in the number of network nodes involved in data processing and transport. For example, the RNC network element is going away and its functionality will be distributed between the new access gateway (AGW) and the evolved node B (eNode B). By 'flattening' the network architecture, developers expect improvements in data latency and real-time communications, as shown in Figure 5.

These architectural changes will move service processing from the RNC to the eNode B – in closer proximity to the access network – in order to reduce communication latency and improve call performance. This means that the eNodeB will execute both control and data plane processing, similar to the scenarios discussed in this section. As demonstrated by the benchmark data, Intel® multi-core processors provide the computing resources required to support both control and data processing in an eNode B network element.

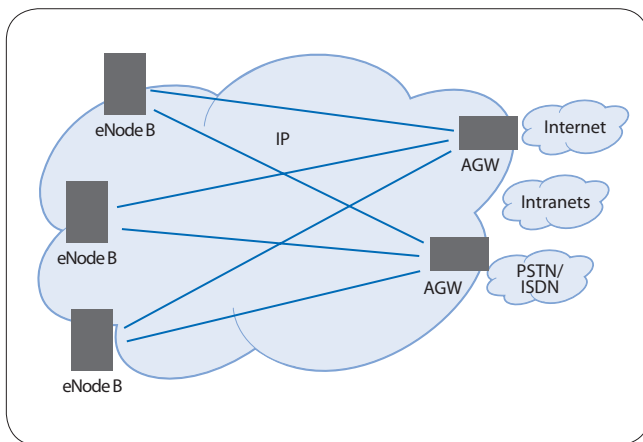


Figure 5. 3GPP Long Term Evolution (LTE) Network.

Flow Processing Prototype

This section provides performance data for a flow processing prototype based on Intel® quad-core processors, shows the performance impact of executing control and data processing on a single system and discusses some useful software optimizations. The flow processing benchmarks demonstrate the ability of Intel general purpose processors to perform both control and data processing on a single board at the same time.

Flow processing is a key function in various packet applications such as security processing, media processing and quality of service (QoS) processing. It classifies packets into flows based on a set of specified parameters and applies a set of operations on those packets. The flow processing pipeline primarily consists of the blocks shown in Figure 6.

The flow processing pipeline is implemented as a Linux kernel module, which bypasses the Linux network stack and receives packets directly from the network driver. The Hash Lookup block takes packets from the RX (receive) block and calculates a hash value that is used to assign packets to one of the CPU cores performing flow processing. This implementation employs the 'jhash' routines available in the Linux kernel, which operates on five fields in the packet: IP Source Address, IP Destination address, IP source port number, IP destination port number and protocol ID.

The Policy Lookup block classifies the flows and determines which rules should be applied to the flows. The Action Handling routine block modifies packets and schedules them for transmission by the TX (transmit) block.

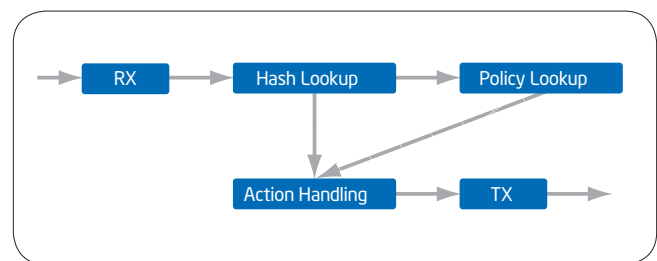


Figure 6. Flow Processing Pipeline.

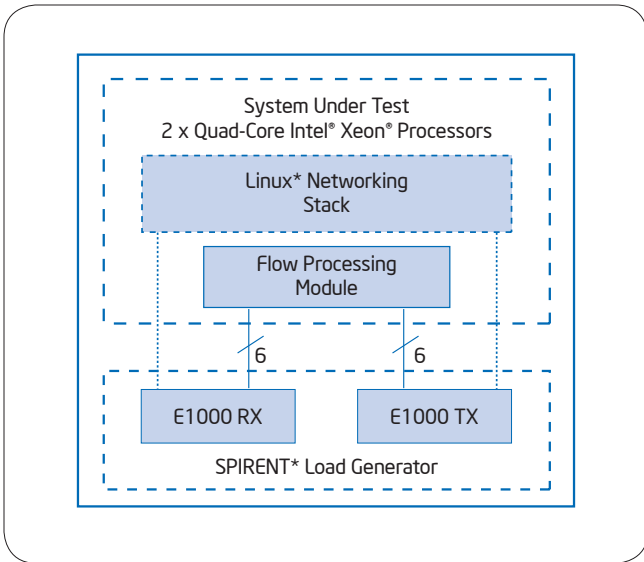


Figure 7. Prototype Test Configuration.

Prototype Test Configuration

A Spirent® Load Generator creates packet traffic for the receive and transmit blocks (RX and TX) using twelve gigabit Ethernet links that are connected to the system under test (SUT), as shown in Figure 7.

The system under test (SUT) is equipped with two 2.66 GHz Quad-Core Intel Xeon processor 5300 series (a total of eight CPU cores) and 12 Ethernet interfaces producing a maximum load up to 12 gigabytes per second (Gbps). During testing, six CPU cores process packet flows while two cores are idle, which allows for the possibility that a networking element could also deploy control functionality. More details about the SUT are provided in the appendix.

Packet Forwarding Test Results

The SUT operates in duplex forwarding mode, where each of six CPU cores handles two gigabit Ethernet networking interfaces operating at full line rate, as shown in Figure 8. These interfaces are assigned to specific cores using symmetric multi-processor (SMP) affinity and Interrupt Request (IRQ) affinity mechanisms. The E1000 driver for Ethernet network adapter employs polling rather than interrupt mode, which reduces the CPU overhead associated with servicing standard interrupts. In this mode, each

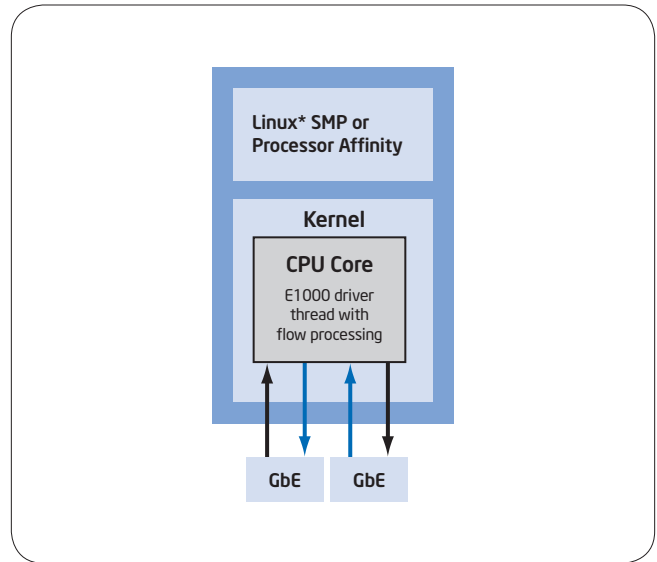


Figure 8. Duplex Forwarding Configuration.

core services 2.8 Mpps (2x1 Gbps) at 64-byte Ethernet frame sizes and performs packet reception, table lookup, header modification and packet transmission.

With six CPU cores active, the SUT processed 8.75 million packet per second (Mpps) without any dropped or lost. The throughput increased as more cores were activated, as shown in Figure 9.^{2,3}

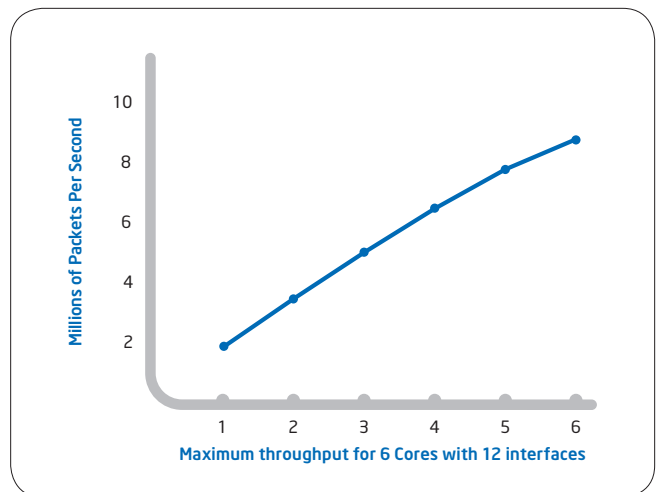


Figure 9. Duplex Forwarding Results.

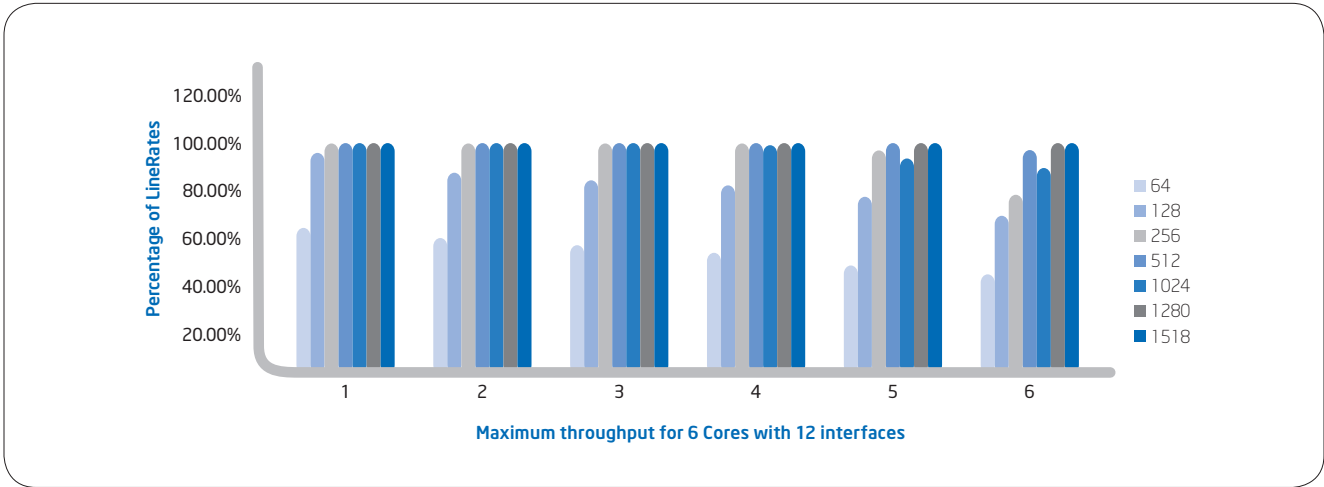


Figure 10. Duplex Forwarding Results By Packet Size.

Next, the load tests were repeated for different packet sizes. The tests were conducted for common packet sizes such as 128, 256, 512, 1024, 1280 and 1540 bytes. Figure 10 shows the maximum throughput based on packet size and the number of active CPU cores.^{2,3} As the number of cores increases, an increase in the aggregate traffic is accompanied by a slight decrease in the percentage of serviceable line rate.

Impact of Control Plane Processing

In the previous example, two CPU cores are idle, leaving open the possibility they could perform control plane processing. This section shows the impact on packet processing when these cores are activated. Control plane processing is typically a compute-intensive operation. Most control plane protocol layers consume incoming packets, step through state machines and update table data structures. To simulate control processing, this prototype runs a software module that reads and writes to a table and performs some calculations between table lookups.

In this test, the size of the table varies between zero and 8 megabytes, corresponding to the total amount of the L2 cache memory in the SUT, as shown in Figure 11. As the control plane table size increases, valuable cache memory resources are diverted from the data plane applications, which reduces flow processing performance. The resulting throughput decreases from 9.1 to 8.5 Mpps, which corresponds to a seven percent performance reduction, a relatively small amount. This demonstration indicates that a single board based on embedded Intel multi-core processors can provide flow processing performance comparable to network processors, while also supporting other applications like control plane processing.

As mentioned previously, the prototype employs a module that reads and writes to a table. All table entries are the size of a cache line (64 byte); they are read randomly and written sequentially. Figure 12 shows the delay, as indicated by the number of CPU cycles, to read a table maintained by the two CPU cores processing the control plane functions.

As the control plane table size increases, the read latency increases. When the cache table size becomes more than the size of one CPU core cache, 4 megabytes, the read latency increases by almost a factor of two. The key learning is that the control plane table should be at most the size of the L2 cache in one CPU core and not the sum of all the cache memory in the system.

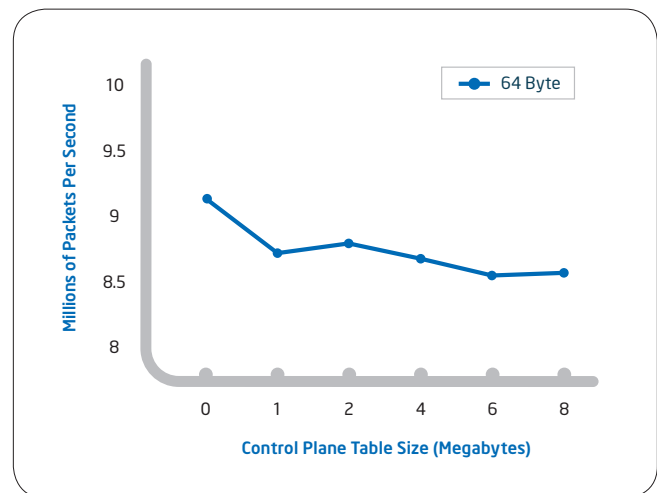


Figure 11. Data Plane Performance versus Control Plane Table Size.

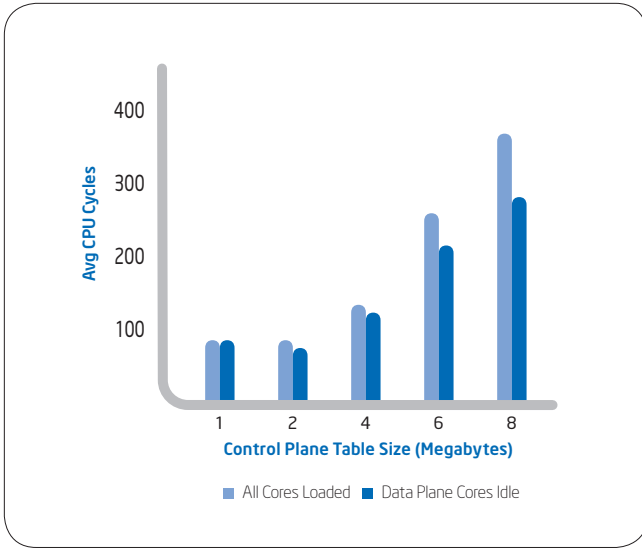


Figure 12. Average Number of Cycles to Read a Table Entry on the Control Plane Cores.

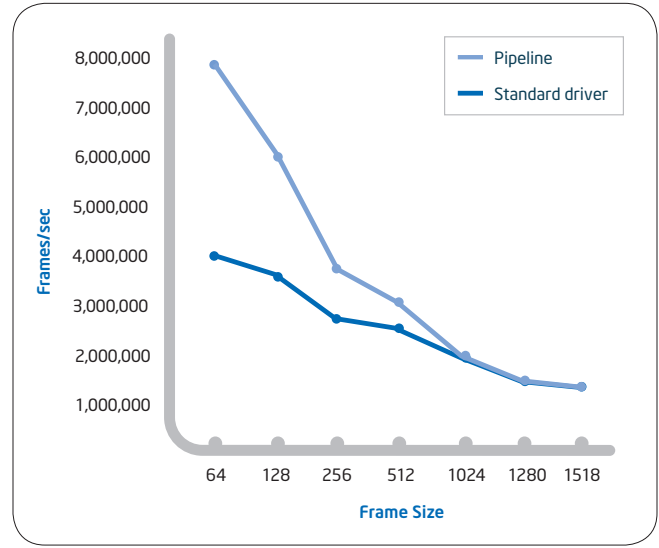


Figure 13. Performance of Standard Driver versus Pipeline.

Packet Processing Performance Optimizations

The flow processing prototype employs application software and driver optimizations which fall into the following categories:

- **Application Specific Optimizations:** In flow processing applications, it's important to minimize the data sharing between CPU cores, which reduces bus traffic and improves cache efficiency. This is achieved by processing all of a flow's packets on one CPU core using a hash function to disperse packets among the available cores. This maximizes locality of reference, a critical factor affecting cache performance.

Pipelining is another technique used to increase CPU core efficiency. The packet processing flow is broken up into different stages, each assigned to a different CPU core. In concept, when each core has a relatively small task, there's a better chance the code remains in cache so it executes faster. If the tasks are unique, data sharing is minimized. The difference between using a standard driver and a pipelined version is illustrated in Figure 13. For small packets, the throughput of the pipelined version is more than two times greater than a standard Linux driver.

- **Infrastructure Optimizations:** There are aspects of the Linux networking infrastructure that are not particularly multi-core friendly. For example, the Linux networking stack is prone to locking, so the prototype bypasses this utility and implements an overall architecture that reduces locking. With the original Linux driver, only one CPU core can receive and transmit at a time, stalling the other cores. This resource locking is eliminated by a queue, which buffers the traffic from the CPU cores and

has full control over the network interfaces, such as receive and transmit operations. As a result, each CPU core can communicate with any interface, unencumbered by locking structures that degrade performance.

- **Minimizing Memory Bottlenecks:** In a multi-core environment, developers try to minimize memory contention issues between CPU cores. Memory utilization is improved by aligning cached data structures, minimizing memory accesses and reducing shared data structures. Another powerful technique is inserting software prefetch instructions into the application software. These instructions pre-load the CPU caches with data that is imminently required by the code path and avoid stalls caused by the CPU cores waiting to access memory.
- **Driver Optimizations:** Performance can be improved by modifying the way drivers update pointers. For example, the network adapter E1000 driver exchanges data with the hardware through a set of descriptor structures that are designed as circular queues and updated through a set of pointers. The driver updates one pointer, and the device updates another. In order to maintain data coherency, the pointers are maintained in un-cacheable memory, so any modification to these pointers is a fairly costly operation. The standard driver updates the transmit descriptor pointer on every packet scheduled for transmission. Instead of updating the pointer for every transmission, the driver should be changed to update the pointer after sending out a few packets. By bundling pointer updates, performance may be improved by as much as ten percent while having minimal impact to latency.

System Partitioning Approaches

There are several generic ways to partition control and data plane applications running on a single board. This section describes para-partitioning, a scheme that offers cost and performance benefits.

System designers partition control and data processing for several reasons. Partitioning allows these applications to be developed and scaled independently, and helps prevent applications from adversely affecting each other. In addition, partitioning helps accommodate special computing environment requirements. For example, control plane applications often run in a general purpose environment (e.g., Linux), whereas data plane applications typically require dedicated resources (e.g., network interface adapters) and perhaps real-time operating systems.

In general, system partitioning is used to 1) consolidate multiple applications on a single platform, 2) tune performance through resource provisioning and 3) isolate applications. When multiple applications are supported on a single board, there are several partitioning schemes worthy of consideration:

Operating System–Based Schemes employ the operating system’s built-in task-scheduling capabilities to assign tasks to specific CPU cores.

Virtual Machines use a hypervisor, also called a virtual machine monitor (VMM), to oversee partitions, called virtual machines (VMs), that run operating systems and applications.

Para-partitioning Methods assign platform resources during system BOOT.

Of these three options, para-partitioning is perhaps lesser known, and the following provides a brief overview.

Para-Partitioning Overview

Para-partitioning creates multiple logical machines on a single board by statically allocating platform resources at startup (system BOOT). Para-partitioning uses the Extensible Firmware Interface (EFI), which is a specification that defines a model for the interface between operating systems and platform firmware. EFI is the follow-on to traditional BIOS implementations for Intel processor-based platforms, and more information may be found at UEFI www.uefi.org.

In a standard multi-core processor-based system, one CPU core is designated as a boot processor and all the other CPU cores are designated as application processors. All boot operations are performed on the boot processor with application processors held in reset. Once the operating system is loaded, it initializes all the CPU cores and runs a scheduler on each of the cores. This sequence is slightly modified to implement para-partitioning.

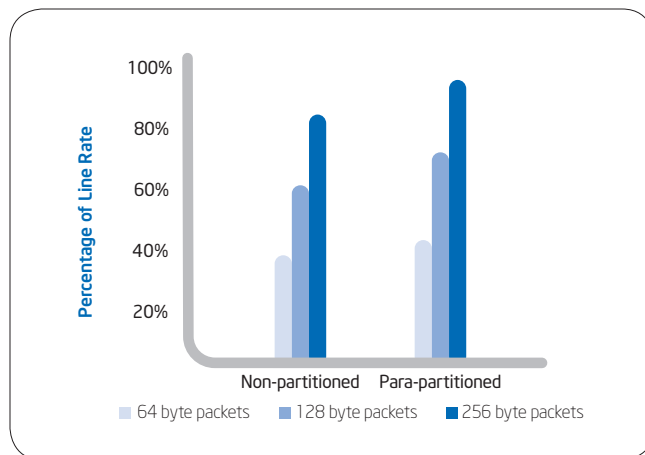


Figure 14. IPv4 Forward Throughput.

In a para-partitioning system, one CPU core in each logical partition is selected as the boot processor. Then, each partition undergoes the bulk of system initialization and execution of drivers that either initialize particular components in the system (e.g., PCI enumeration) or initialize a service, such as a serial port interface. Each partition executes this phase independently and multiple instances of EFI are created, each of which will behave as an independent logical system. Each of these logical systems can load and boot different operating systems.

Advantages of Para-Partitioning

Para-partitioning offers a low-cost, performance-oriented means to create logical partitions. There’s no middleware to license and the software actually runs on the bare hardware, so it’s fast. There are no hypervisors or other mediating software layers controlling access to the hardware. Para-partitioning allows for tight control over resources and the application environment.

In contrast, virtual machines access hardware resources through the VMM, which introduces a performance penalty, especially for applications servicing a large number of interrupts. Operating system-based partitioning schemes also introduce overhead, such as scheduling and resource management, and can hinder developers who require precise control over software execution. In addition, the impact of control and data processing on communications is not completely understood for applications with kernel-intensive operations, especially since kernel pre-emption should be tuned-off.

Para-partitioning can also improve packet processing performance. In Figure 14, Internet Protocol Version 4 (IPv4) forwarding throughput is tested for three packet sizes. When the system is para-partitioned with two logical partitions with four CPU cores each, it runs between 8 and 15 percent faster than non-para-partitioned, depending on packet size^{2,3}

Conclusion

With Intel multi-core processors, system developers have the option to combine control and data plane functions on a single board and eliminate packet processing boards, which lowers cost and speeds up content processing. Developers can deploy packet processing applications using standards-based components such as Linux, high-performance Quad-Core Intel® processors and off-the shelf hardware components. The benchmark results presented in this paper show that standards-based solutions have performance comparable to platforms based on dedicated hardware solutions, such as the Intel IXP2800 network processor. They also offer many software benefits, such as enabling a code base that runs on a single platform, performance scaling through additional CPU cores, and industry leading development tools.

Multi-core architecture is now the vehicle for delivering greater computing performance, yielding processor families that offer higher levels of scalability and more predictable performance. The roadmap for embedded Intel processors provides developers with many more cores in the future to further increase system capacity and throughput. These processors are backed by robust development tools and technical assistance from Intel that enables the rapid development and optimization of software applications.

These products are also supported by the Intel® Embedded and Communications Alliance, one of the world's most recognized embedded ecosystems. The alliance provides equipment manufacturers with a trusted supply line of scalable Intel®-based products and services, and offers leading-edge products, design expertise and total life cycle support from worldwide service organizations. This rich ecosystem of products and services helps developers speed time-to-market and improve reliability.

Appendix

1) Definition of RNC Terms

AGW (Access Gateway) – An LTE network element that connects eNode B elements to IP-based network infrastructure.

eNode B (evolved Node B) – An LTE network element that performs traditional Node B services plus 3G RNC data path function.

GGSN (Gateway GPRS Node) – Provides a gateway interface to external Packet Data Networks (PDN).

ISDN (Integrated Services Digital Network) – A wide area network (WAN)-oriented data communication service provided by telephone companies.

LTE – Long-Term Evolution which refers to next-generation wireless architecture to succeed 3G.

MSC (Mobile Switching Center) – A network element that connects wireless systems to circuits-switched and ISDN networks.

Node B – A network element responsible for radio communications between cells within a service area and mobile handsets.

RNC (Radio Network Controller) – A network element responsible for controlling the use and the integrity of radio resources.

PSDN (Public Switched Digital Network) – A publicly available network supporting packet-switched data, separate from PSTN.

PSTN (Public Switched Telephone Network) – The traditional telephone system that serves the world's public circuit-switched networks.

SGSN (Serving GPRS Node) – Provides voice and packet data services and management of mobile subscribers.

2) System Configuration

The system under test is comprised of two Quad-Core Intel® Xeon® processor 5300 series (a total of eight CPU cores), the Intel® E7520 Chipset and the Intel® 6300ESB I/O Controller Hub, as shown in Figure 15. The system memory is configured with two gigabytes of DDR2 memory.

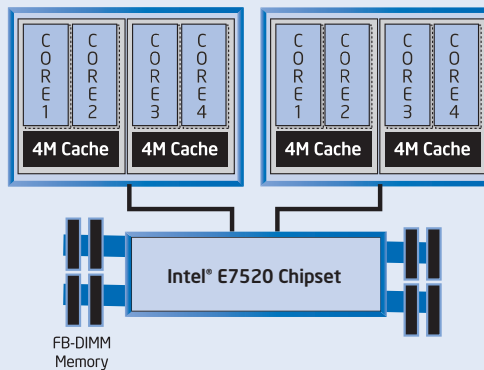


Figure 15. Quad-Core Intel® Xeon® processor 5300 series-based platform.

¹ Based on data from Nielsen//NetRatings, International Telecommunications Union, worldinternetstats.com and Intel analysis.

² Performance tests and ratings are measured using specific computer systems and/or components and reflect approximate performance of Intel® products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit http://www.intel.com/performance/resources/benchmark_limitations.htm

³ For additional information about the benchmark results described in this paper, please contact your local Intel Sales Representative.

*Other names and brands may be claimed as the property of others.

Copyright © 2008 Intel Corporation. All rights reserved.

Intel, the Intel logo, Intel Core, Xeon, and Intel NetStructure are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

