Technology Guide

Intel® Dynamic Load Balancer (Intel® DLB) - Scaling of IPsec Workloads

Authors

Niall McDonnell

Declan Doherty

Pravin Pathak

Rashmi Shetty

Abhishek Khade

Mrittika Ganguli

1 Introduction

Pipelined multi-stage software architecture provides the scalability that a conventional run-to-completion (RTC) model fails to provide. A Network Adapter queue, or a port processed by a single core, can be a bottleneck, limiting the per-flow processing rate for an application. To achieve scalability, applications can be separated into stages, and each stage is executed in parallel. Queues and arbiters can connect pipelined stages and perform complex scheduling and load balancing tasks. In such situations, Intel® Dynamic Load Balancer (Intel® DLB) can be used as a hardware queue manager and load balancer to offload queuing and scheduling tasks from software.

To illustrate this concept, the Data Plane Development Kit (DPDK) IPsec Gateway sample application was chosen because it experiences performance limitations related to the RTC model in the presence of high-bandwidth tunnels. The application was modified to support a pipelined design to provide scalability, while the DPDK Eventdev library manages queuing and scheduling. The implementation was tested on both the software event device and Intel® DLB.

This document is part of the Network Transformation Experience Kits.

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	February 2023	Initial release.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
DPDK	Data Plane Development Kit
DUT	Device under test
Eventdev	DPDK Event Device
cldemote	Cache Line Demote instructions
Intel® DLB	Intel® Dynamic Load Balancer (Intel® DLB)
IO	Input/output
IPsec	Internet Protocol Security
PoC	Proof of Concept
RSS	Receive Side Scaling
RTC	Run To Completion
SA	IPsec Security Association
SP	IPsec Security Policy
SQM	Software Queue Manager used for software-based load balancing
Tunnel	Refers to IPsec security tunnel

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Elephant Flows	https://en.wikipedia.org/wiki/Elephant_flow
Event Device Library	https://doc.dpdk.org/guides/prog_guide/eventdev.html
Queue Management and Load Balancing on Intel® Architecture White Paper	https://builders.intel.com/docs/networkbuilders/SKU-343247-001US-queue-management-and-load-balancing-on-intel-architecture.pdf

2 Overview

2.1 Scaling and Performance Challenges

Figure 1 shows a simple representation of a multi-engine RTC IPsec implementation. This simplified representation has IO at both ends with several workers in between to process the IPsec workload. The IO devices have several options to distribute the load across available processing engines (local cores/workers). This distribution is typically achieved by using the Network Adapter RSS feature that distributes traffic based on a generated hash (typically computed using fields from the packet header). The Network Adapter distributes traffic over multiple receive queues that are mapped to processing engines. This strategy guarantees atomic processing of the traffic flows by always mapping them to the same engine. The main challenges to high throughput IPsec workloads are efficient scaling and fair load distribution.

2.2 Elephant Flows

An elephant flow is a high bandwidth traffic flow that consumes a significant portion of (or even exceeds) the processing capability of a single thread. Such flows may be limited in number but can take up a significant share of the network bandwidth.

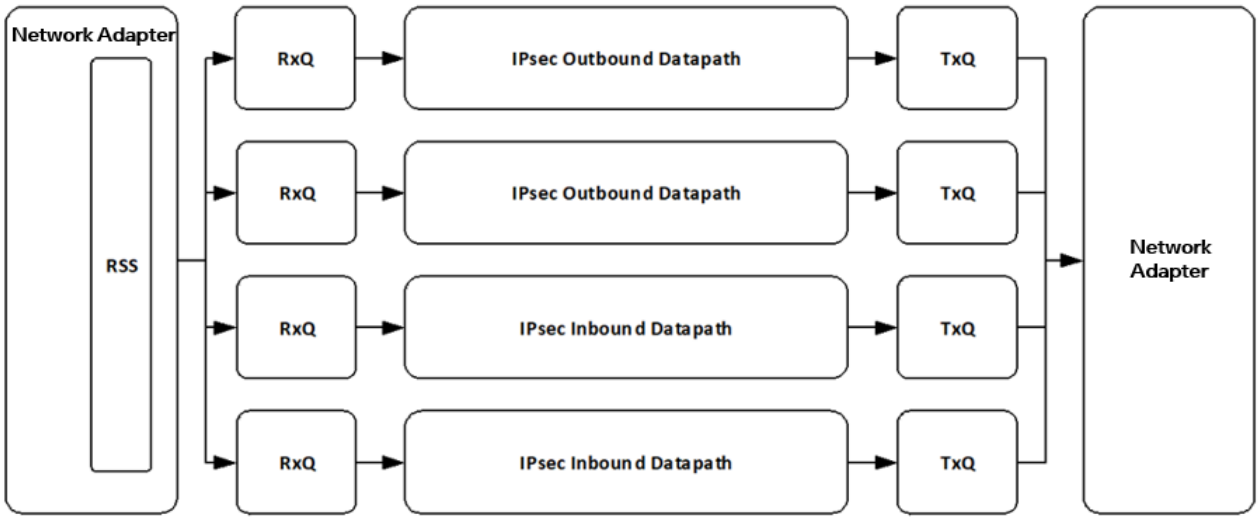


Figure 1. Logical Representation of a Multi-Threaded Platform

2.3 IPsec Scaling

Unless there is sufficient entropy in arriving traffic, the application is unlikely to distribute it efficiently using static, hash-based distribution mechanisms. Furthermore, a single core becomes the bottleneck for maximum per flow throughput that can be achieved. With a pipelined model, processing is distributed across multiple stages, and can scale even in the presence of poor entropy and high bandwidth elephant flows.

For a pipelined model, there are different outbound and inbound requirements.

3 Data Plane Development Kit (DPDK) Eventdev Library

Similar to Ethdev or Cryptodev APIs, DPDK has the Event device library that adds support for an event driven programming model. It offers applications automatic multi-core scaling, dynamic load balancing, pipelining, ingress packet order maintenance and synchronization services to simplify application packet processing.

By introducing an event driven programming model, DPDK can support both direct polling (without using Eventdev) and event driven programming modes for packet processing. Applications are free to choose the model (or combination of the two) that best suits their needs.

For example, the ordered scheduling of Eventdev allows an application to parallelize tasks while maintaining ingress order without any synchronization mechanisms needed inside the application. Eventdev's atomic scheduling sends a flow of events to a particular core, which preserves flow order and allows software to operate atomically without requiring any locks. It also load-balances flows to new cores when flows are not actively being processed.

Similar to the Ethdev and Cryptodev libraries, Eventdev has a poll-mode driver (PMD) architecture. An Eventdev can be a hardware device, in which case the PMD programs the device, or a software implementation, where the PMD implements the event scheduling logic. Since the Eventdev API presents a common interface for all its PMDs, an Eventdev application is portable across a wide variety of platforms. For more details on the Eventdev library, refer to the Event Device Library section of the DPDK programmer's guide.

3.1 Eventdev Applications

Applications based on Eventdev architecture use a pipelined model as shown in Figure 2. Producer and consumer stages are needed if the Network Adapter is not able to send traffic directly to the Eventdev. Individual workers then make up the processing stages of the pipeline. This architecture is dynamic and can scale up or scale down the number of workers based on the required system throughput. The producer handles receiving packets from the Network Adapter interface and enqueues them to the Eventdev device for load balancing. The consumer is the final stage of the pipeline and handles the aggregation of traffic from all the workers and transmission to the Network Adapter. Figure 2 shows a simple logical representation of this architecture with two pipeline stages. It is typically the same set of workers that implement multiple pipeline stages.

3.2 Hardware Eventdevs

Intel has introduced the Intel® Dynamic Load Balancer (Intel® DLB) device, which is a hardware accelerator that offloads multi-core packet scheduling and load balancing. It supports all required DPDK Eventdev features and more. Eventdev PMD for Intel DLB is part of DPDK since release 20.11 and is compliant with required Eventdev APIs.

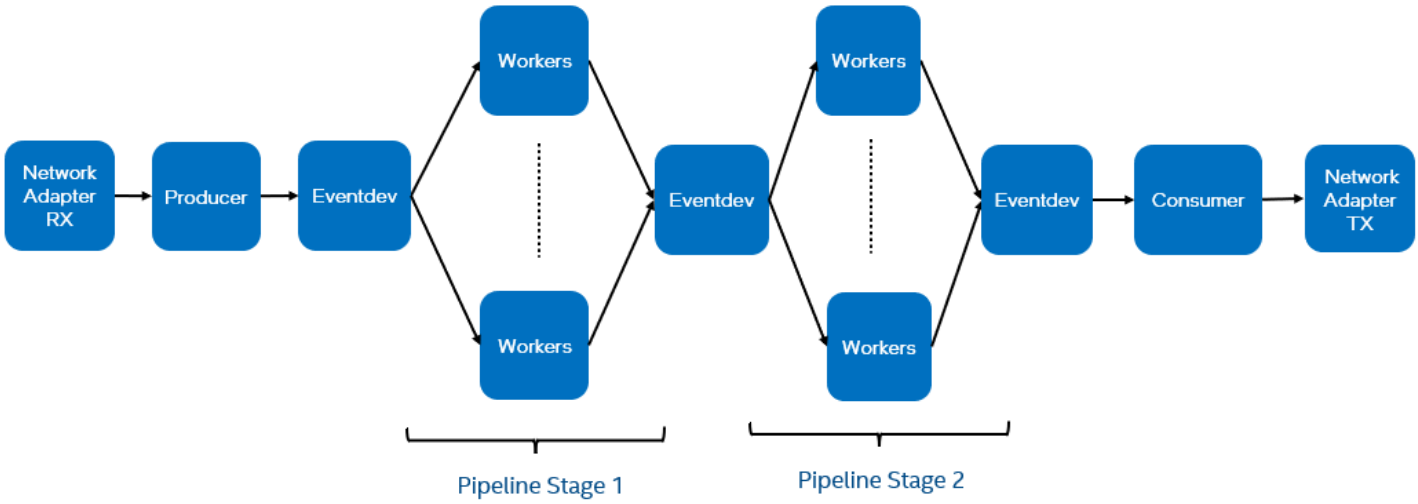


Figure 2. Pipelined Model for Applications Based on Eventdev Architecture

Intel DLB is a hardware implementation containing a system of queues and arbiters connecting event producers and consumers. Intel DLB is a PCI device in the CPU package, and it interacts with software running on the cores and potentially other devices. The Intel DLB load-balancing features include:

- Lock-free multi-producer/multi-consumer communication
- Multiple priorities for varying traffic types
- Various scheduling types such as atomic, ordered, and parallel

For more details on Intel® DLB, refer to the Queue Management and Load Balancing on Intel® Architecture white paper.

4 IPsec Gateway Application

In this proof of concept (PoC), both outbound processing and inbound processing is divided into three stages each. Outbound processing is divided into classification, sequence number allocation and encryption plus route stages. Sequence number allocation must be atomic per Security Associations, while other stages should be ordered. Inbound processing is divided into inbound classification plus anti replay check, decryption plus route and anti-replay update stages. Both anti replay check and update stages are atomic since the sequence numbers check and window update need to be done atomically on a per SA basis. The decryption and routing is an ordered stage. Both the encryption stage and the decryption stage take advantage of load balancing to provide the scaling that a pipeline design provides.

The work performed in each stage is chosen such that the cross-core snooping is minimal. For instance, the design separates the classification stage from the encryption/decryption stages. The classification stage acts on the packet header and not the actual packet payload, whereas the encryption and decryption work on actual packet data. These stages work on the independent parts of the packet, and as far as possible, they try to avoid touching the same cache lines to reduce cross-core snooping. The application can take advantage of load balancing flows for encryption and decryption stages while keeping the sequence number stage atomic. Similarly, anti-replay check and update stages are separated from the decryption stage.

The IPsec pipeline uses Intel DLB through the Eventdev APIs. Each worker is assigned an Eventdev port at setup. Each worker port is linked to all event queues. Each event queue represents a pipeline stage with its own queue type depending on the stage that it feeds (atomic, ordered). A worker can process packets for all the stages of the inbound pipeline and outbound pipeline. After dequeuing a batch of packets from the port, the worker groups the packets based on their pipeline stage ID (indicated by the associated event queue ID) and processes each packet group based on the pipeline stage ID. After completion, the worker enqueues the entire batch of packets to Eventdev with each packet routed to its own next processing stage.

4.1 IPsec Datapath Pipeline

Outbound and inbound datapaths are divided into three stages as shown in Figure 3. The RTC model runs all of these stages on the same core and in sequence, whereas the pipelined model can run them in parallel on different cores.

4.2 Producer/Consumer Stages

Producer and consumer stages serving the Network Adapter Rx and Tx interfaces are the single entry and exit points for the packets. Therefore, these stages should be designed carefully not to become the bottleneck for the entire pipeline. A producer should be able to feed the pipeline from the Network Adapter at its processing rate and a consumer should be able to transmit at

the same rate back to the Network Adapter. To achieve this, minimal processing is performed at the producer and consumer stages. Producers tend to use larger batch sizes and consumer load is reduced by performing routing in the previous stage with results saved in the packet buffer.

Running producer and consumer threads as hyperthreaded buddies sharing a physical core offers better performance due to sharing of L2 cache.

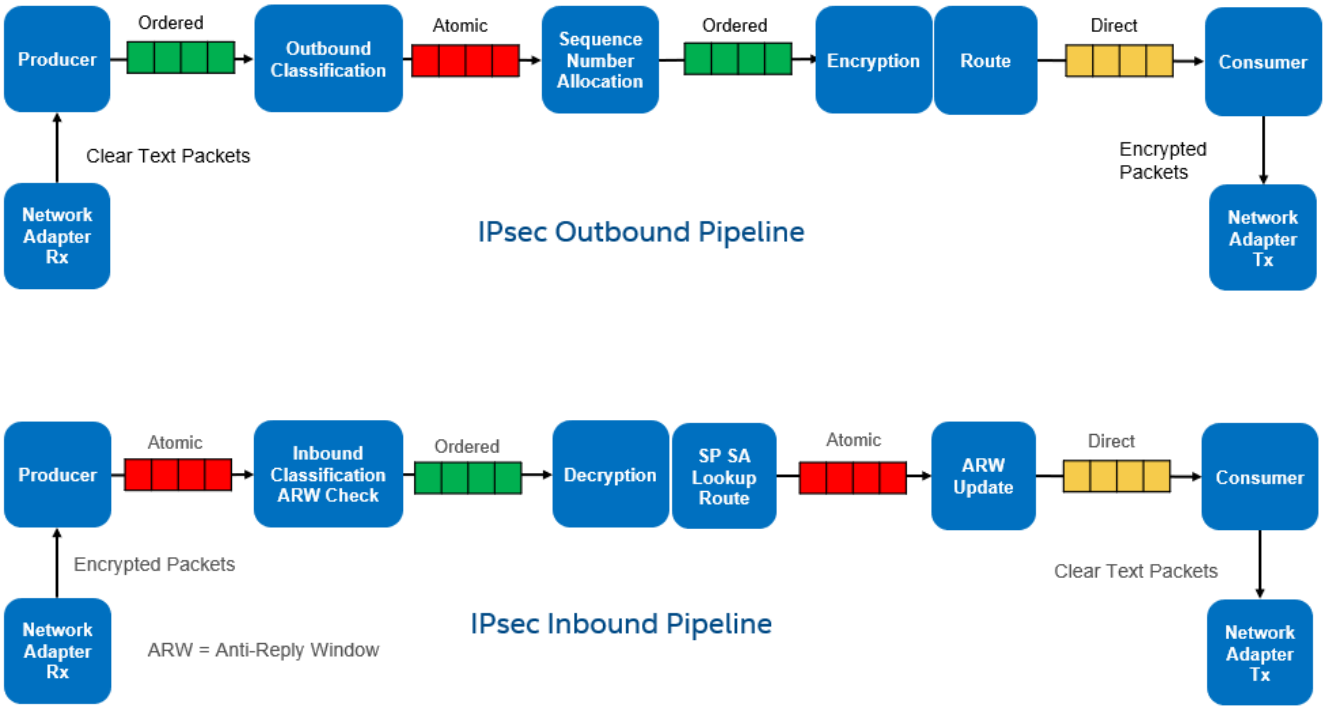


Figure 3. Outbound and Inbound IPsec Datapath Processing Pipeline

4.3 Flow Ordering (Atomic/Ordered) Stages

An atomic queue type guarantees that each atomic flow ID is processed by a single core at a time and the packet order is preserved. As IPsec needs atomicity at the SA level, the SA value is used as the atomic flow ID for all atomic stages in this implementation.

The PoC also uses an mbuf header to pass the SA ID, routing information, sequence number, and so on, between pipeline stages for later stages to use them without recomputing.

Ordered processing stages can load balance packets across multiple workers. These packets are processed in parallel and original enqueue order is restored before sending them to the next stage. Reordering traffic in software is costly and consumes CPU cycles. The ordered queue type supported by Intel DLB performs reordering in the hardware, thereby reducing the cost of CPU cycle.

4.4 Cross Core Data Movement

Cross-core snooping is a major challenge in the pipeline design versus the RTC model. The PoC pipeline stages are designed to minimize cross-core snooping by not accessing the same cache lines in different stages. The VTune™ analyzer identifies hotspots. The Cache Line Demote (*cldemote)* and prefetch instructions are used to minimize the performance impact.

5 System Configuration

5.1 Test Setup

Figure 4 shows the test setup used for this PoC. Two DUTs running pre-production Intel® Xeon® Scalable processors with integrated Intel DLB hardware are used. DUTs are interconnected with a 100 Gig Ethernet link. An IXIA traffic generator with 100 Gig ports generates clear text traffic. DUT1 encrypts the traffic coming from IXIA port-1 and sends it to DUT2 inside the IPsec tunnel. DUT2 decrypts the traffic and forwards it back to IXIA port-2. A similar path is followed by the traffic from IXIA port-2 to port-1. Traffic is always bidirectional for this testing. Testing was performed by Intel in March 2022.

Table 3. Hardware and Software Configuration

Hardware Configuration	Description
Platform	Intel Reference Platform (code named Archer City)
CPU	Pre-production 4th Gen Intel® Xeon® Scalable processor @ 1.8Ghz
Memory	256GB (8x32GB 4800MT/s) - Dual Rank
Storage	1x 240G Intel® SSD
Network Adapter	Intel® Ethernet Network Adapter E810-CQDA2 for QSFP (rev 02) 2x100 GbE link
Turbo	Disabled
Microcode	0x8d000260

Software Configuration	Description
Workload	DPDK* IPsec with Intel® Dynamic Load Balancer (DLB) and Software queue management (SQM)
OS	Ubuntu* 20.04.2 LTS
Kernel	5.4.0-40-generic
Workload Version	DPDK* IPsec -20.08
Compiler	gcc 9.3.0
CPU Utilization (active cores)	100%
Test date	11 January 2022
Test by	Intel® Network Platform Group

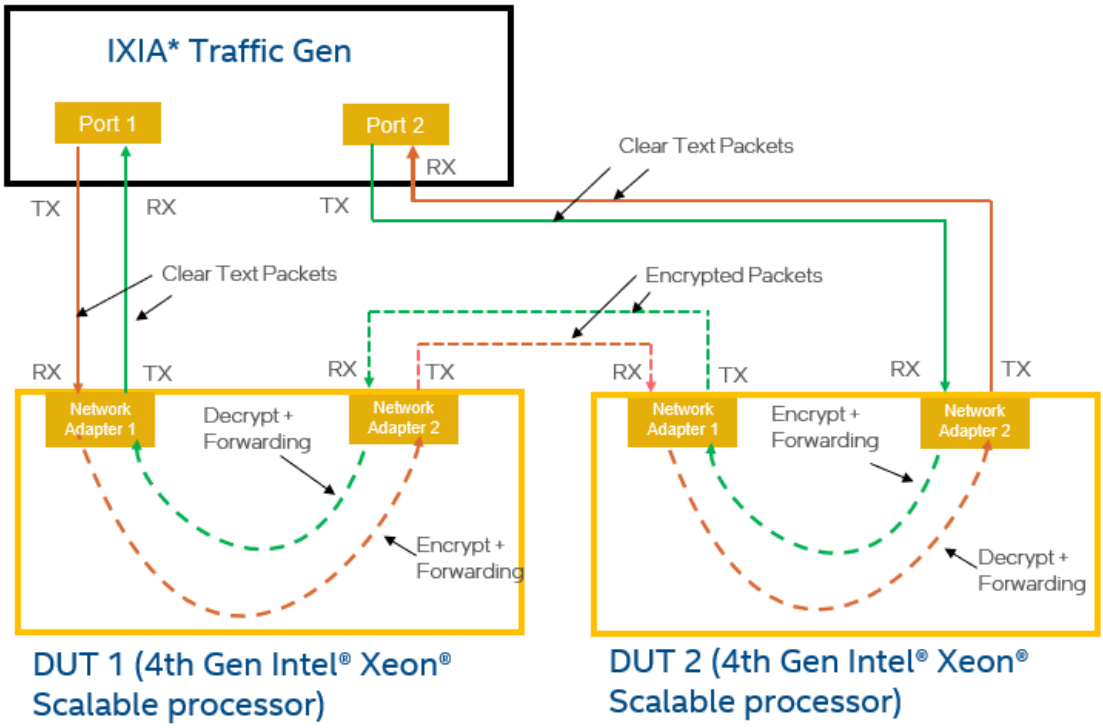


Figure 4. Test Setup

5.2 DPDK IPsec Security Gateway Sample Application

The pipelined IPsec application is an enhancement to the ipsec-secgw application in DPDK v20.08. The application was modified to use the DPDK Eventdev APIs to connect various pipeline stages. It leverages the DPDK IPsec library that was modified for multi-stage support. The security gateway sample application is not a production ready implementation and is designed only to demonstrate the core data path components. It does not support control path integration.

6 Test Traffic Profiles

6.1 Single-Flow Test Traffic

The single-flow traffic test is designed to test the scalability of an IPsec implementation with a single, high-bandwidth IPsec SA that has traditionally been a performance bottleneck for IPsec implementations.

Performance of a single 64-byte packet flow, viewed in terms of the number of packets per second that the architecture can support, demonstrates the implementation scalability with respect to protocol handling, classification, and forwarding. The large packet (>1 KB) performance for an IPsec SA, viewed in terms of throughput, demonstrates the capabilities of the implementation to handle scaling of cryptographic workloads.

6.2 Multi-Flow Test Traffic

A test with multiple, high-bandwidth traffic flow demonstrates the ability of the implemented architecture to handle flow distribution of IPsec workloads. This test focuses on a small number (<32) of flows, as many flows are easy to distribute. This is also a scenario that has been difficult to load balance with static load distribution.

7 Performance

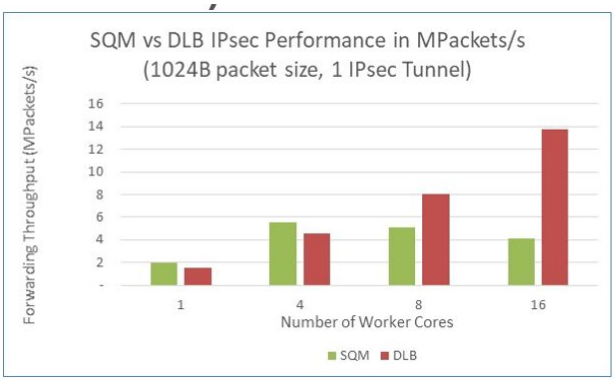
This section describes the performance characteristics of the defined IPsec datapath reference architectures, and it highlights the relative performance and limitations. It also demonstrates the value of Intel DLB in enabling effective load balancing and scaling of IPsec workloads.

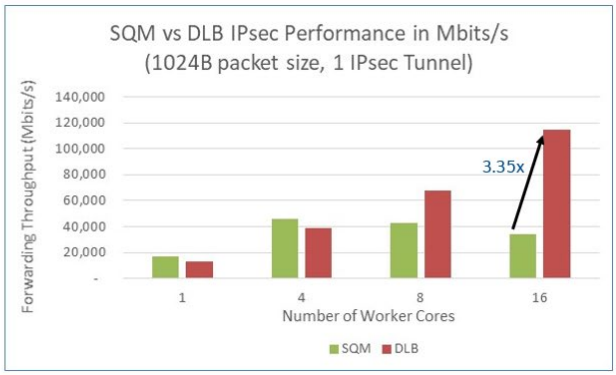
7.1 Run-to-Completion

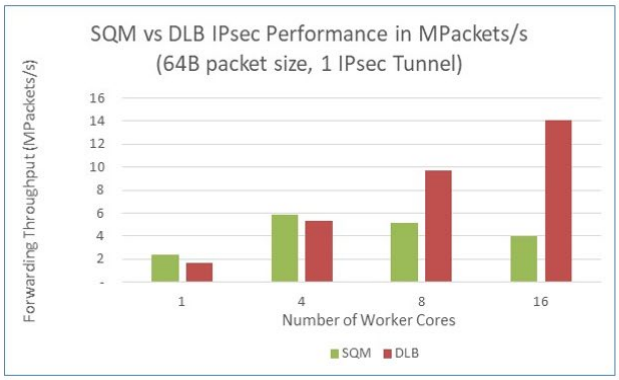
The run-to-completion design of the original DPDK sample saturates early with lower packet rates depending on a single core processing capability.

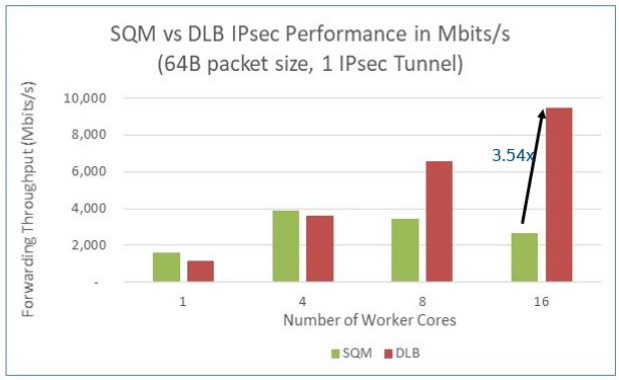
7.2 Pipelined Model (Event-Based)

This is the main architecture discussed in this paper and implemented as a PoC. It was tested using SQM (software Eventdev poll mode driver) and Intel DLB poll mode driver (PCI device). When SQM is used, one additional core is needed to run the SQM scheduler. In this setup, the SQM scheduler could perform a maximum of 15 million scheduling decisions per second.









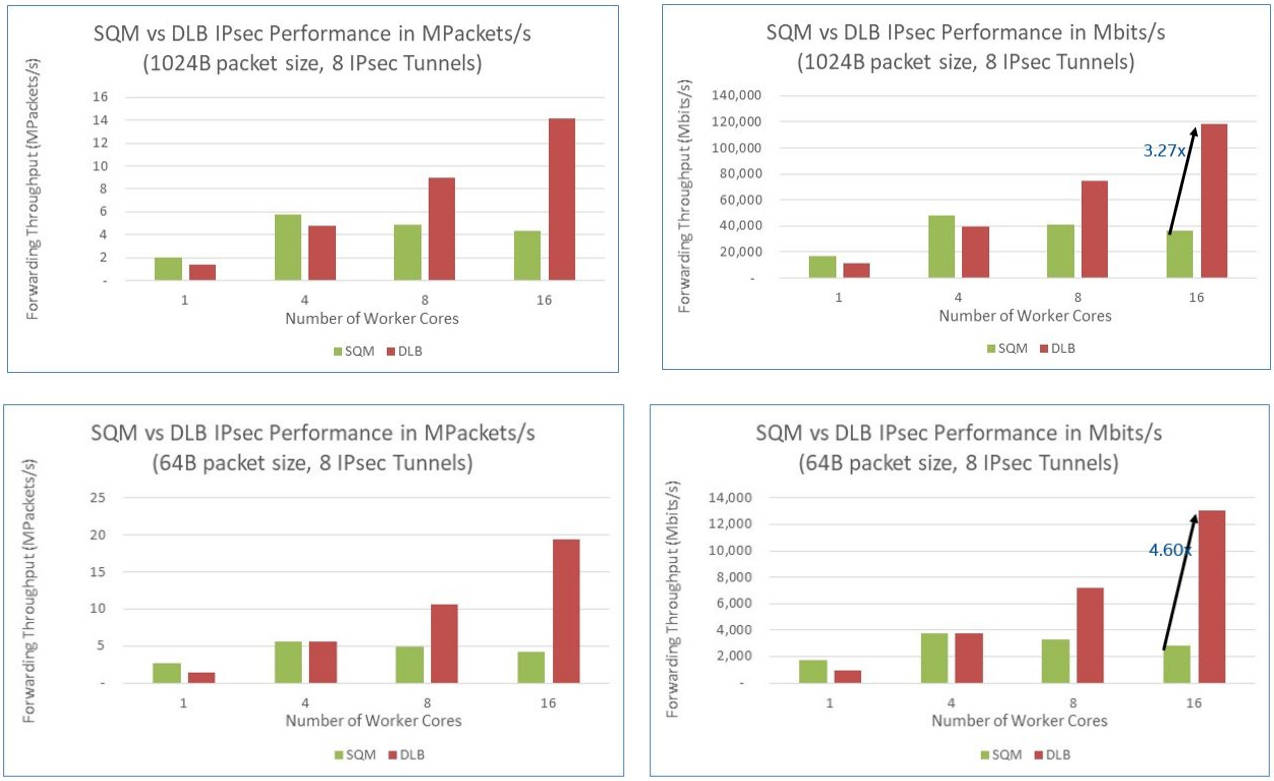


Figure 5. MPackets/s, Mbits/s Packet Performance Results (Top to Bottom) 1024B, 1 Tunnel; 64B, 1 Tunnel; 1024B, 8 Tunnels; 64B, 8 Tunnels

8 Summary

In the RTC models, the maximum flow bandwidth that can be processed is restricted by the processing capacity of a single core. This PoC shows how this limitation can be overcome using a multi-stage pipeline. The pipeline design needs queues to connect various stages and complex scheduling and load balancing. When implemented in software, this task is an overhead and at times saturates at lower bandwidths. This PoC shows how to overcome this limitation by using the Intel DLB hardware accelerator. The DPDK Eventdev library is a proven, easy way to manage multi-stage connectivity that allows simple integration of different hardware and software accelerators.

0223/DN/WIT/PDF 767663-001US