Silicom SPDK Overview

Introducing Extension and Auxiliary Libraries for Intel® DPDK

1. High Level Description

1.1 Overview

Silicom Performance Development Kit (SPDK) is a set of libraries, service and configuration files targeted to simplify the adaptation of Intel® DPDK to customer's applications.

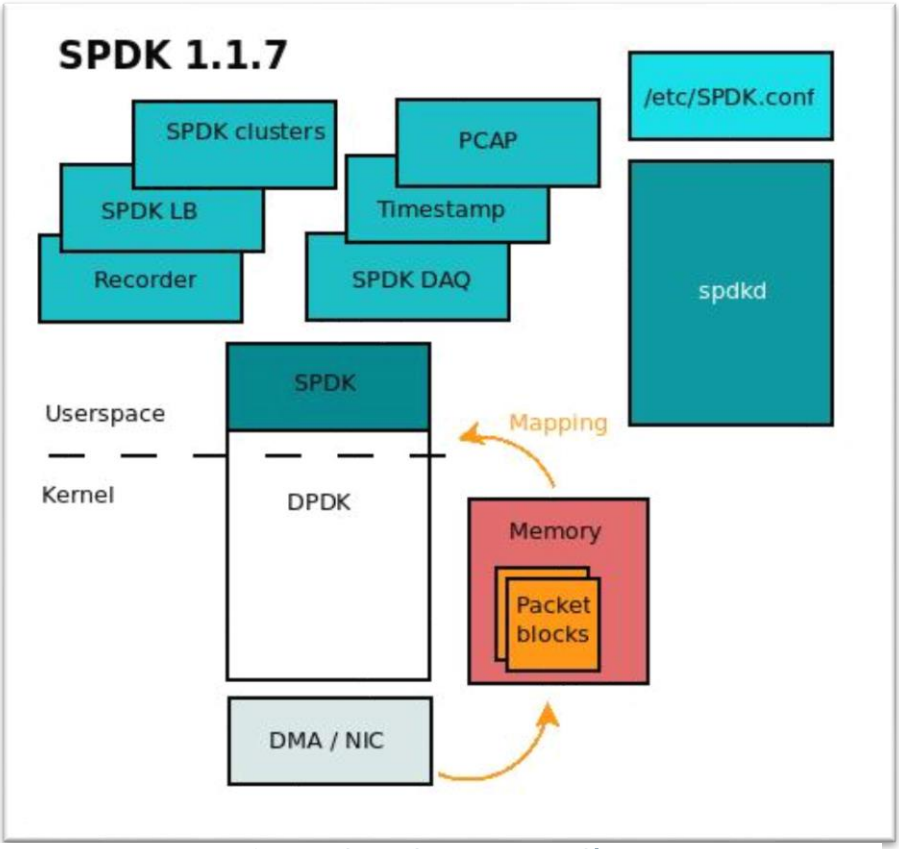


Figure 1 - SPDK Components at a Glance

As seen in figure 1, SPDK includes the following components:

- SPDK daemon – a system service process, instantiated at system boot up, and is responsible for environment setup. Most important task of this service huge pages allocation, memory reservation, file system mount, device driver installation, etc. This daemon secures the largest contiguous memory regions, and repeatedly seeks for better regions allocation during runtime. The key benefit of this service steps from the fact that it preserves system memory for all subsequent SPDK based applications; thus insures system survivability.
- /etc/SPDK.conf – Static configuration file for memory management, license key, etc.
- SPDK API – Rx and Tx base line API, statistics API, files API, etc.

- Memory management – Packet block level work. Packets are held on block structures that are ringed together. Working with packet blocks, rather than on by-packet basis, can increase overall packet processing.
- Recorder – Line rate packet recording to disk utility.
- Clusters – clustered thread API for multi-threaded processing with.
- SPDK LB – Ingress frames load balancing either by software or assisted by hardware.
- PCAP – Support for .pcap format for capture files.
- DAQ – Snort data acquisition component for incoming traffic.
- Timestamp – Support for Silicom hardware timestamp adapters.

1.2 Design goals

Robustness and survivability – Intel® DPDK is a kernel bypass mechanism, and as such, it involves within it a lot of system management aspects such as memory management and handling, core affinity, and many other aspects that are far beyond the core business logic of what process that eventually would run over DPDK. Therefore, SPDK strives to enable the programmers to focus on their application business logic, while keeping the other infrastructure chores, like memory management, as a system service.

API upscale – Intel® DPDK's API is wide, elaborate and confusing in many cases. SPDK brings a simplification to this API.

Better performance – SPDK brings with it set of features and samples that demonstrate a technique to work with DPDK, that in many cases performs better than standalone DPDK. Packet blocks and clusters are examples for that.

1.3 Design Benefits

The goals of the design of SPDK bring several benefits, out of which the most prominent are summarized in the following table:

Memory Recovery	Acheived through SPDK service
	• SPDK daemon starts at boot up and allocate (and reallocate) contiiguous hugepages • Configuration marameters are kept persistent in /etc/spdk.conf • Subsequent SPDK processes do not access OS directly for memory allocation
Linear Scalability	Acheieved with cluster threads
	• Ingress traffic is load balances accross SPDK threads • As more threads affinitied to more cores, the more performence gained • Scaling up with more cores usage as well as with more CPU sockets usage
Jumbo Frame	Acheived with packet blocks and packet segments
	• Packets are kept in memory segments • Packet that is larger than a segment is stored in more than one segment • An array of segment forms a packet, and array of packets froms a block

Table 1 - SPDK Benefits

The benefits that stem from design therefore can be summarized in robustness, flexibility and scalability.

2. Threads Clusters

2.1 Overview

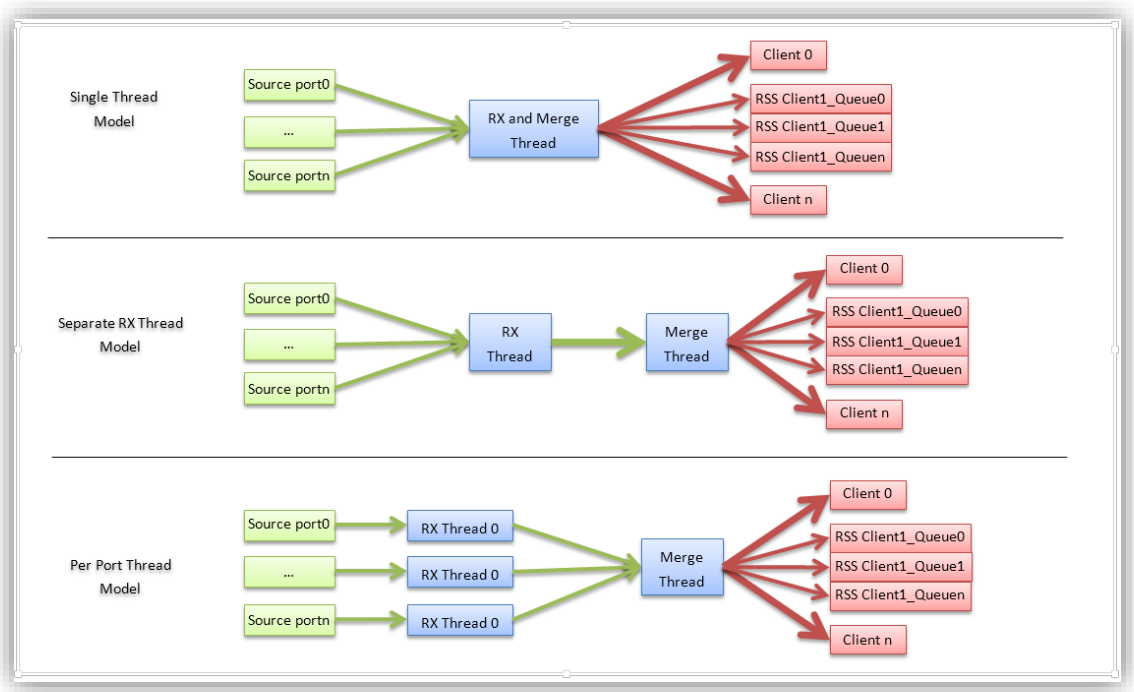
SPDK offer a scalable cluster API. This API enable quick yet flexible multiple processing through threads instantiation, with Rx queues association, per thread. This way, a balanced processing scheme for ingress traffic can be achieved.

2.2 Models

There are three types of models of DPSK clustering, as described in Figure 2. On single thread model, traffic may be sourced from multiple ports, and distributed to several processing entities ("clients"). This mode can be referred as a starting point for application development above SPDK threads cluster API, where subsequently, an optimization may be performed by moving to of the other modes.

On separate Rx thread model, one thread is dedicated for receiving streams of packet, while another thread merges traffic and fans it out for processing threads. This mode enable better utilization of CPU cores.

Figure 2- SPDK Threads Clustering Models



The third mode, the most advanced mode, spawns discrete Rx thread per ingress port, while merging and packet distribution is performed by yet another separate thread.

2.3 Packet Distribution

There are two modes for packet distribution of incoming traffic. First one is hardware assisted and the other is performed by software.

2.3.1 Receive Side Scaling (RSS)

Taking advantage of Silicom's Intel® based adapters to enqueue ingress traffic according to five tuple optimized hash, SPDK threads can be spawned to accept and process the slice of the traffic that is distributed in a balanced (and symmetric if needs be) manner.

2.3.2 Software Load Balance

In case where ingress traffic distribution is required to be done based on other parameters than five tuple, a software based hash can be devised by the application, in order to split traffic across processing entities. This way, non-standard encapsulations, or such encapsulations that otherwise are not supported by hardware, can be identified, be the base for load balance. Examples for that can be stackable MPLS labeling, or GTP tunneling.

3. Roadmap

3.1.1 Bridging

As of DPDK release 2.0, bridging features start to part of the official release, with link bonding capability as a first in a row of features. This same release include also support for Intel® FM10000 series.

As a result of the above, Silicom's focus, as an advanced acceleration technologies supplier, is to offer integration with its Intel® based adapters and accelerators, for layer 2 bridging and switching.

3.1.2 TCP/IP stack

Adding TCP/IP stack capabilities is offered by Silicom through an integration of SPDK cluster threads with DPDK KNI mechanism, using the native OS TCP/IP stack. The idea is to leverage the work of L3/L4 termination, and the direct queues and load balanced ingress traffic, in an intelligent manner, to a socket based interface.