SDC 18
September 24-27, 2018
Santa Clara, CA
www.storagedeveloper.org

Deflate your Data with DPDK

Fiona Trahe & Lee Daly
Intel

Agenda

- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

Agenda

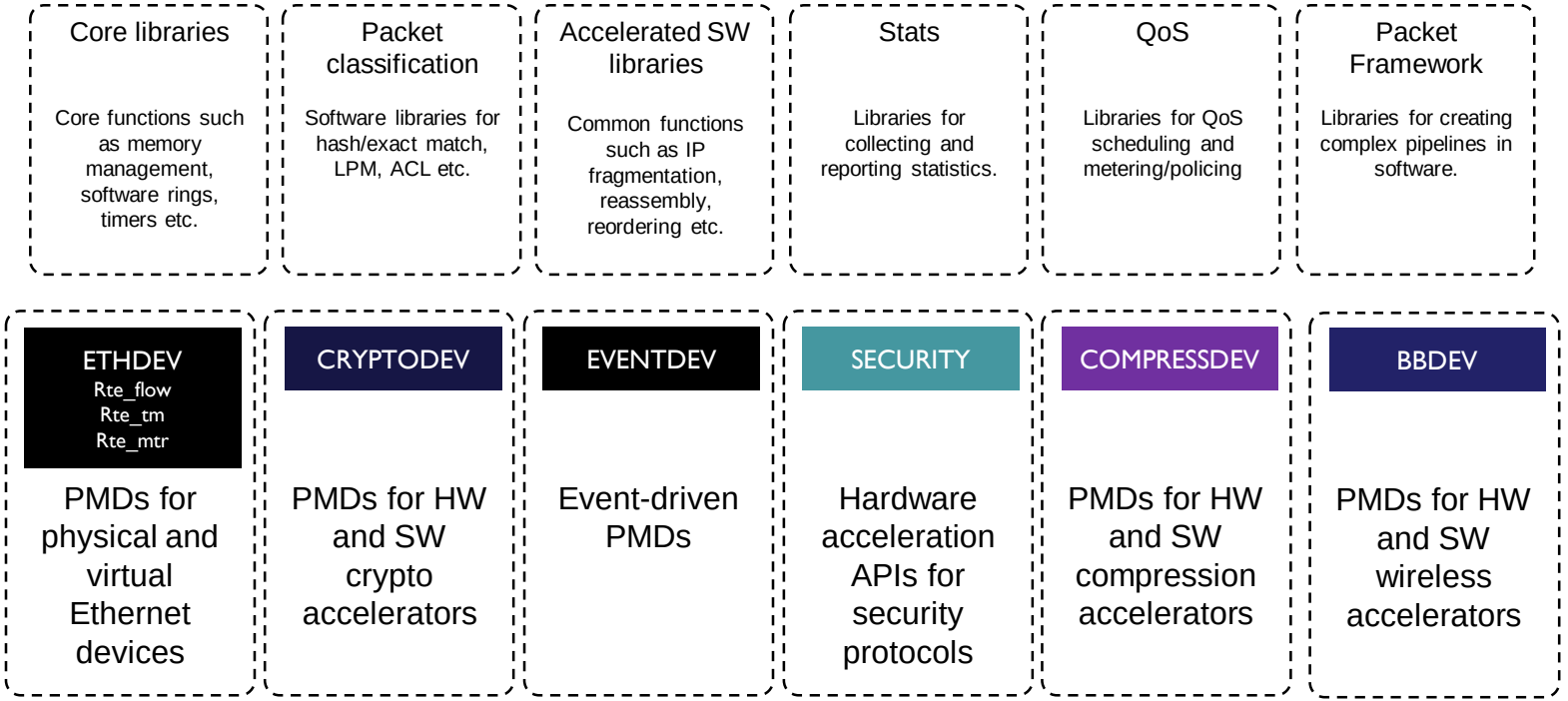
- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

DPDK and SPDK overview

- DPDK https://www.dpdk.org/ is the Data Plane Development Kit that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures.

- SPDK http://spdk.io/ provides a set of tools and libraries for writing high performance, scalable, user-mode storage applications. It relies on DPDK's proven base functionality to implement its memory management.

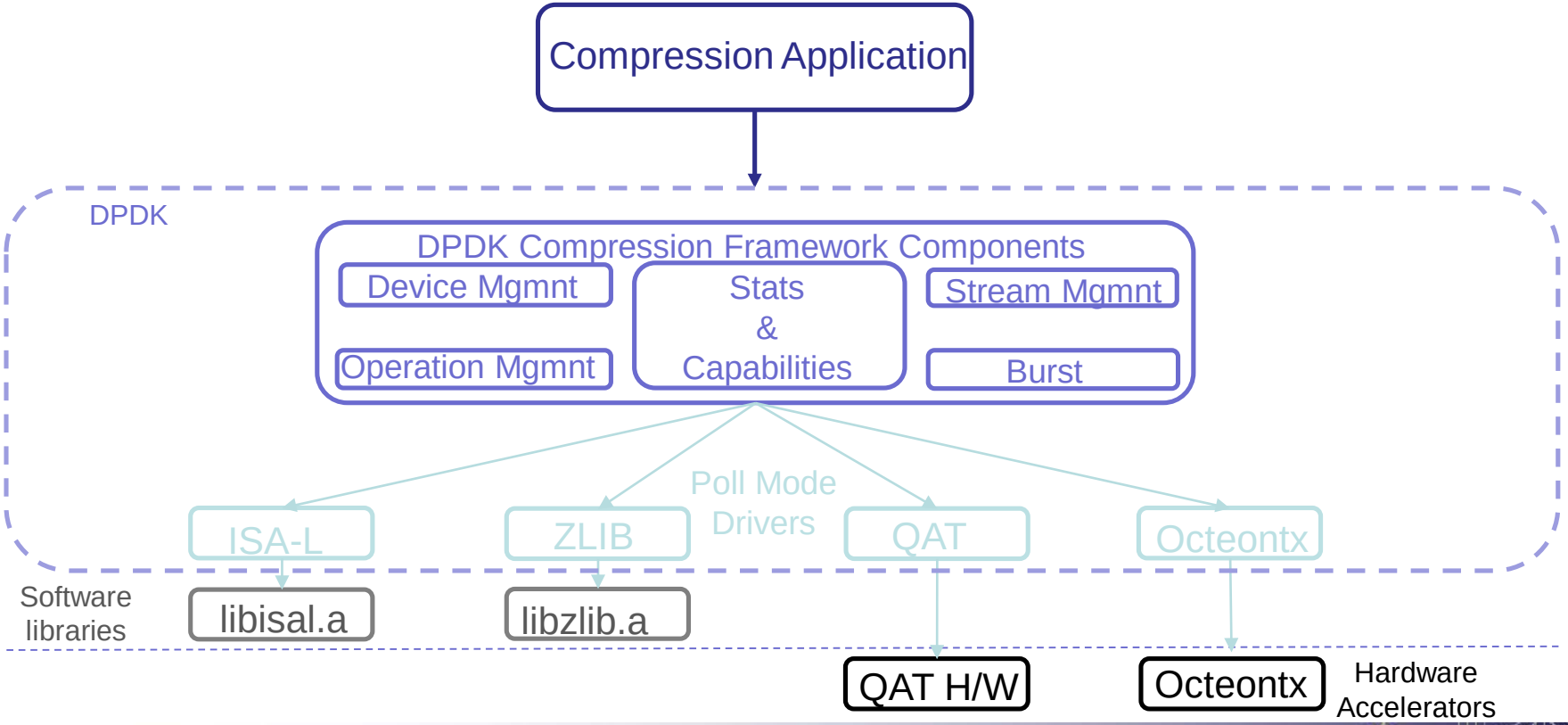
DPDK libraries



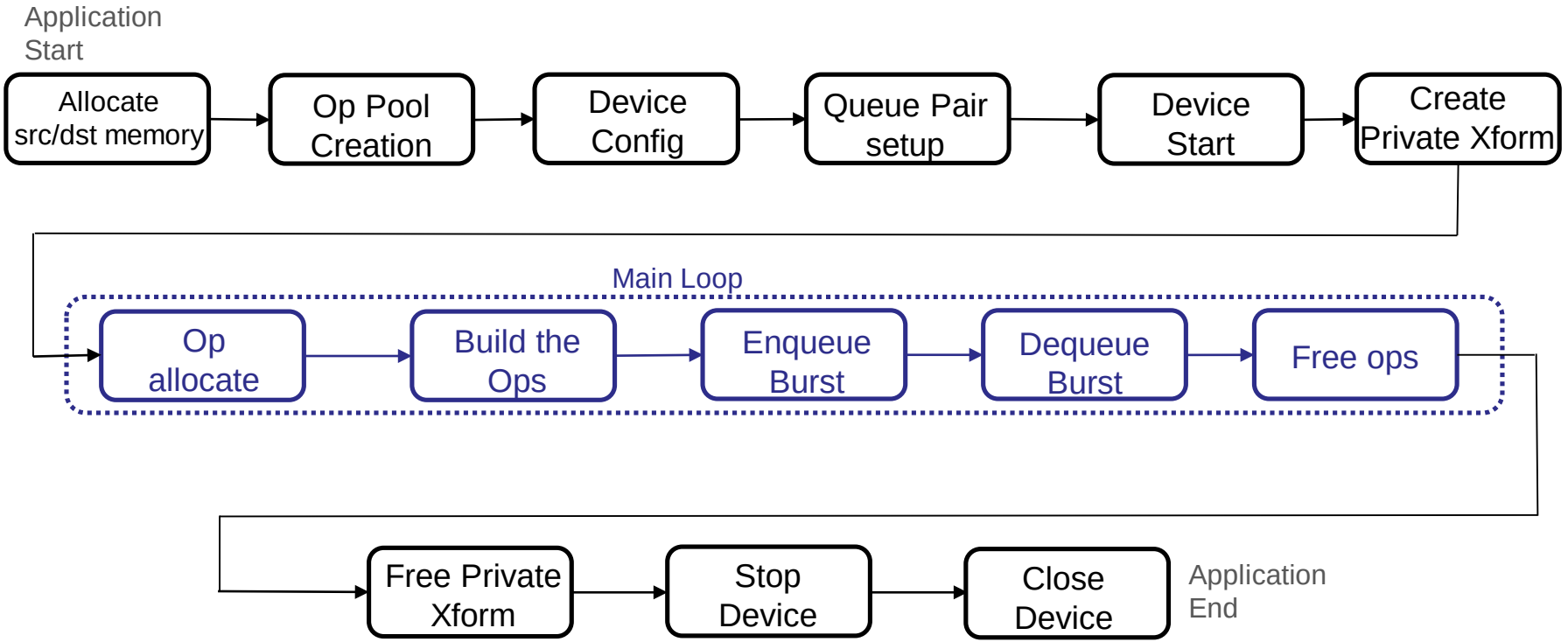
dpdk/compressdev key features

Asynchronous burst API	Chained Mbufs	Compression Algorithms	Compression Levels	Checksum	Hash Generation
To support HW & SW acceleration	To allow compression for data greater than 64K.	Deflate LZS	-1: PMD Default 1: Fastest . . . 9: Best Ratio	#1 CRC32 #2 Adler32 #3 Combined - Adler32_CRC32	#1 SHA1 #2 SHA256

Compressdev Components



Compression API Flow



So you want to run a compression app?

Download and setup DPDK

#1
git clone
http://dpdk.org/git/dpdk

#2
./usertools/dpdk-setup.sh
Option=[22/21]
num hugepages= [64], [64]

Compile DPDK

#3
meson [build directory]

#4
cd [build directory]

#5
ninja

Run Compressdev Unit Test

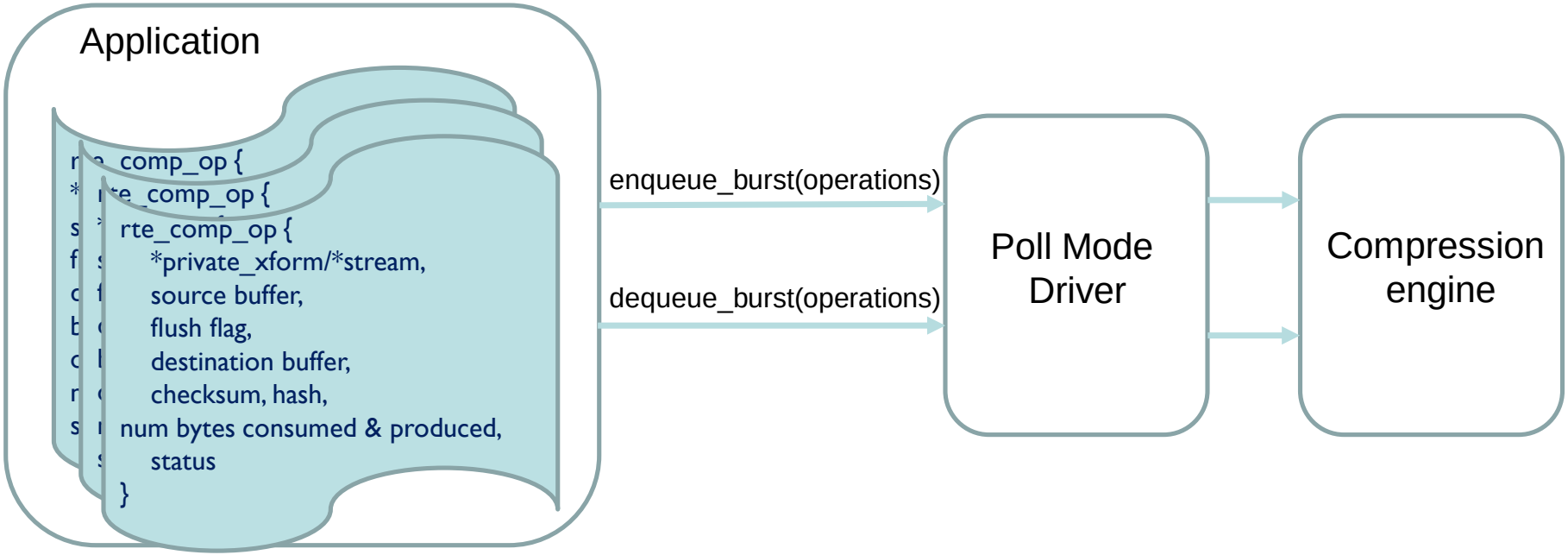
#6
./[build directory]/test/test/dpdk-test
--vdev=compress_isal

#7
RTE>> compressdev_autotest

Agenda

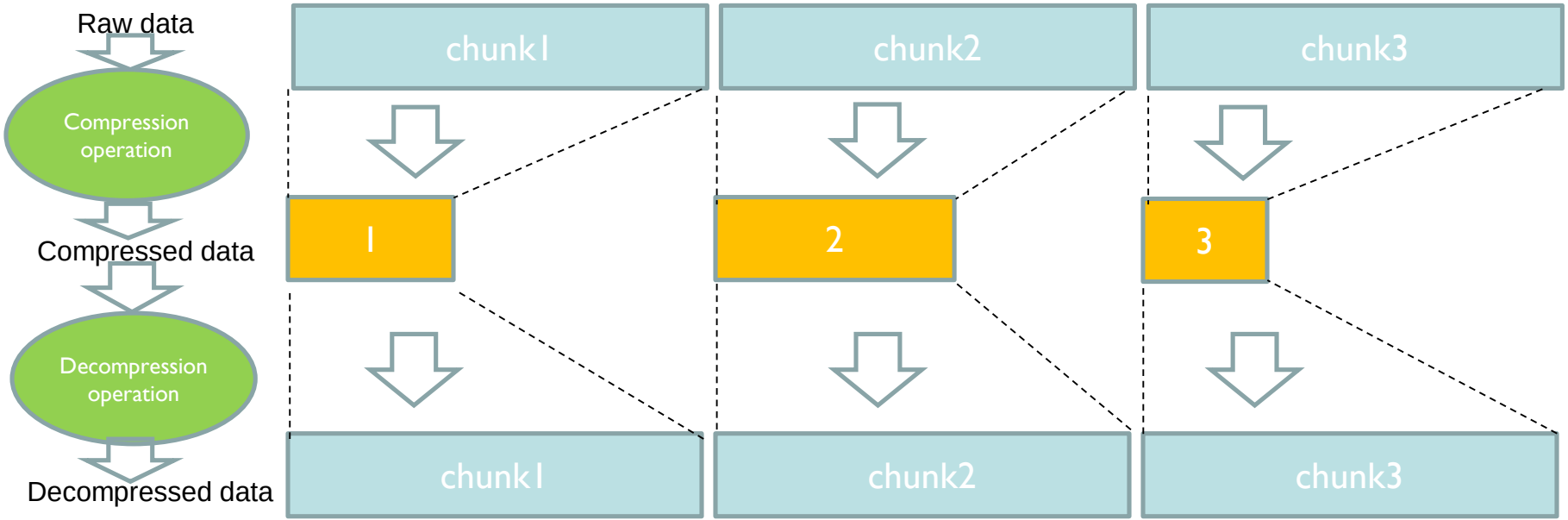
- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

compressdev – operation

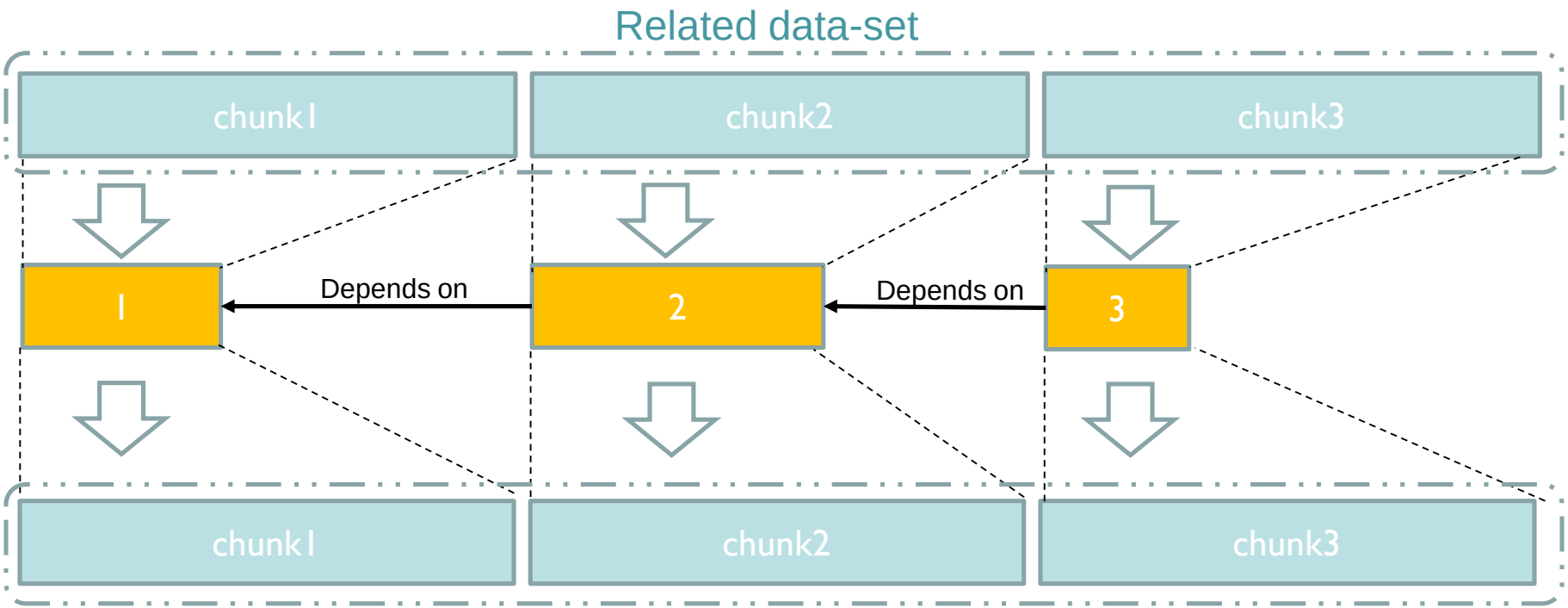


Operations contain both input and output parameters

Compression terminology - stateless



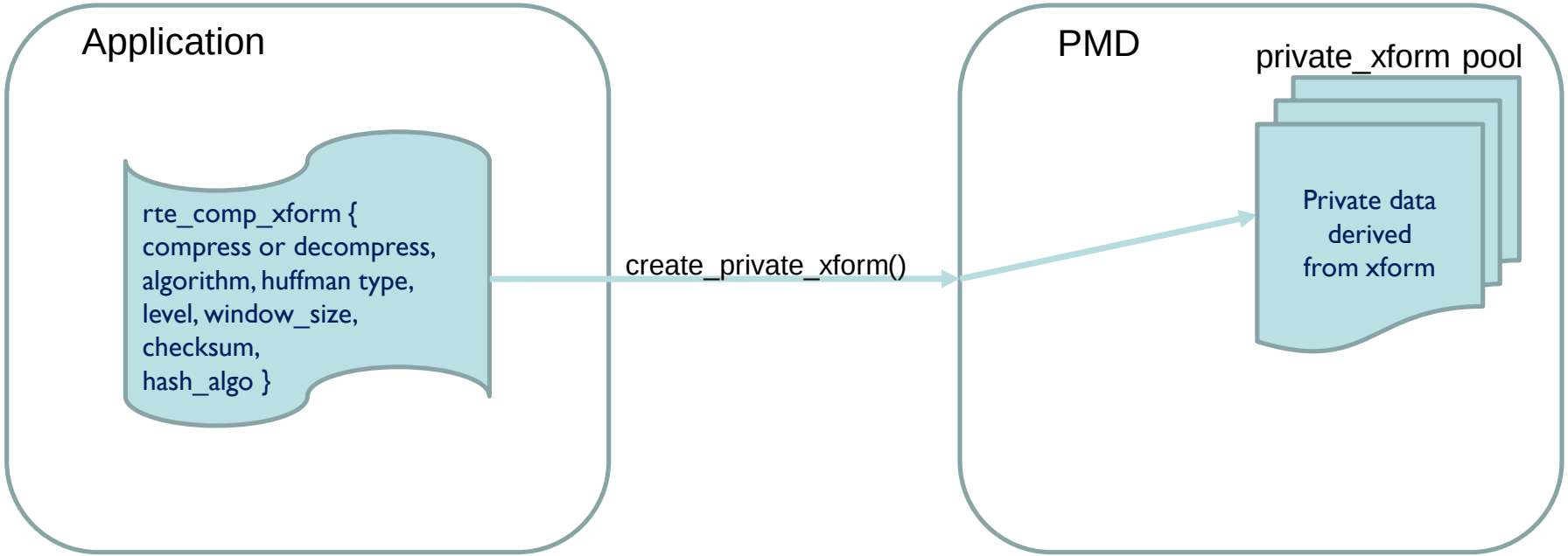
Compression terminology - stateful



Compression terminology recap

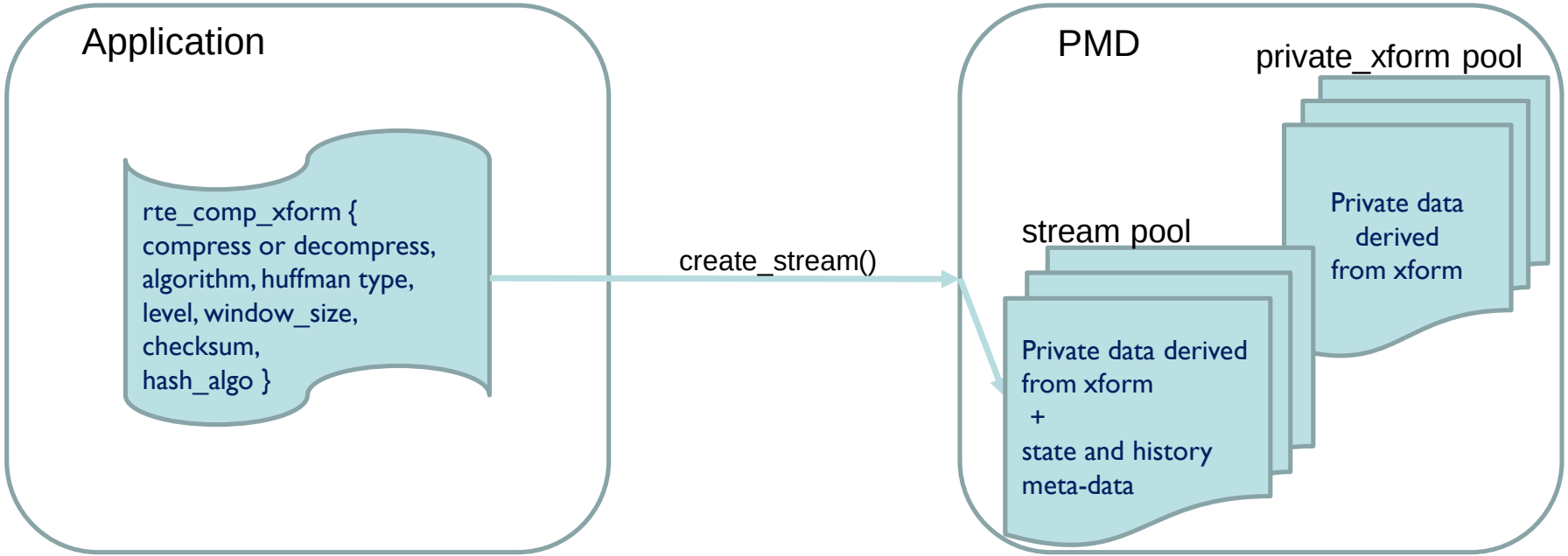
- Stateless compression
  - Data in each operation is treated independently.
  - Operations can be carried out in parallel
  - Each chunk of compressed data can be independently decompressed.
  - Order is not important.
  - May give higher throughput.
- Stateful compression
  - Data in later operations may depend on data in earlier operations
  - History and state is saved at end of each operation to be used in processing subsequent operations.
  - Can only do one operation at a time.
  - Order is important
  - Is useful if all the data is not available at the same time, e.g. related chunks are being received from a network.
  - May give better compression ratio

compressdev – private_xform



A private_xform is used for STATELESS operations

compressdev – stream



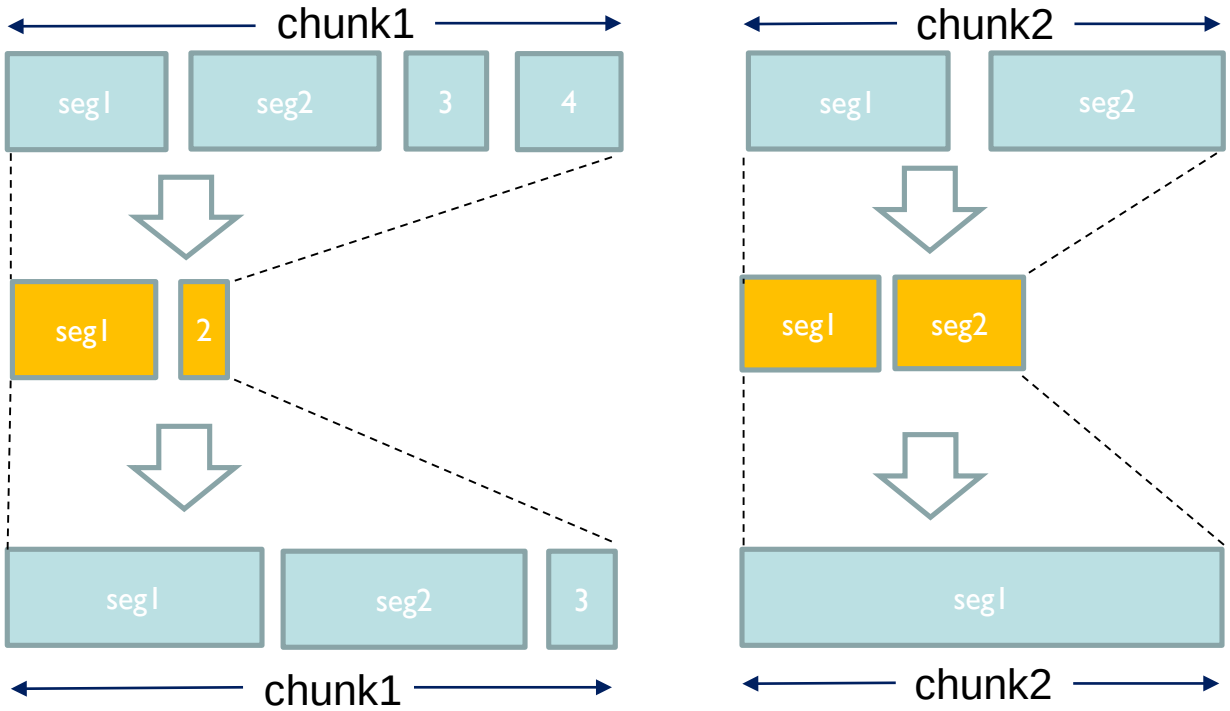
A stream is used for STATEFUL operations

Where do SGLs fit in?

Don’t confuse stateful with scatter-gather lists (SGLs), also called chained mbufs.

- Stateless/Stateful refers to data relationship across operations.
- SGLs are chained data buffers within an operation.
- SGLs can be used in stateless or stateful ops

Compression – stateless with SGLs



- A chunk passed in or out of an operation may be comprised of one or more buffers (segments) chained together.
- Segments can be any size.
- There is no correlation between the number of segments passed in for compression and the number of segments it will decompress to.

Capabilities

Device capability flags

RTE_COMPDEV_FF_HW_ACCELERATED
RTE_COMPDEV_FF_CPU_SSE
RTE_COMPDEV_FF_CPU_AVX512
etc

Compression service capability flags

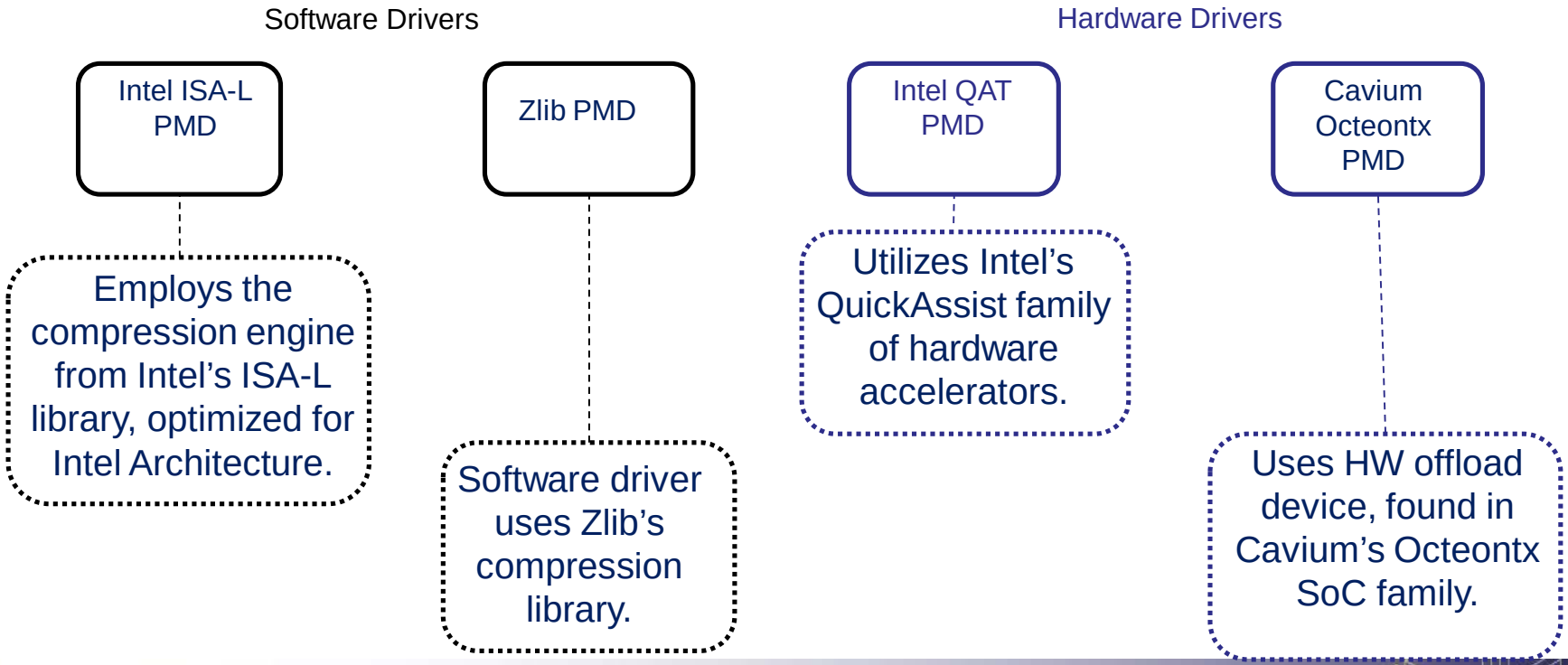
RTE_COMP_FF_STATEFUL_COMPRESSION
RTE_COMP_FF_ADLER32_CHECKSUM
RTE_COMP_FF_CRC32_CHECKSUM
RTE_COMP_FF_SHAREABLE_PRIV_XFORM
RTE_COMP_FF_HUFFMAN_DYNAMIC
etc

```
struct rte_compressdev_capabilities {
    enum rte_comp_algorithm algo;
    /* Compression algorithm */
    uint64_t comp_feature_flags;
    /**< Bitmask of flags for compression service
    features */
    struct rte_param_log2_range window_size;
    /**< Window size range in base two log byte values
    */
};
```

Agenda

- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

compressdev poll mode drivers



Agenda

- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

ISA-L PMD Features

"Deflate" compression algorithm

Adler32 & CRC32 Checksum available

Stateless functionality

Levels

#1 Fastest
#2 Higher ratio
#3 Best Ratio
(AVX512/2 Only)

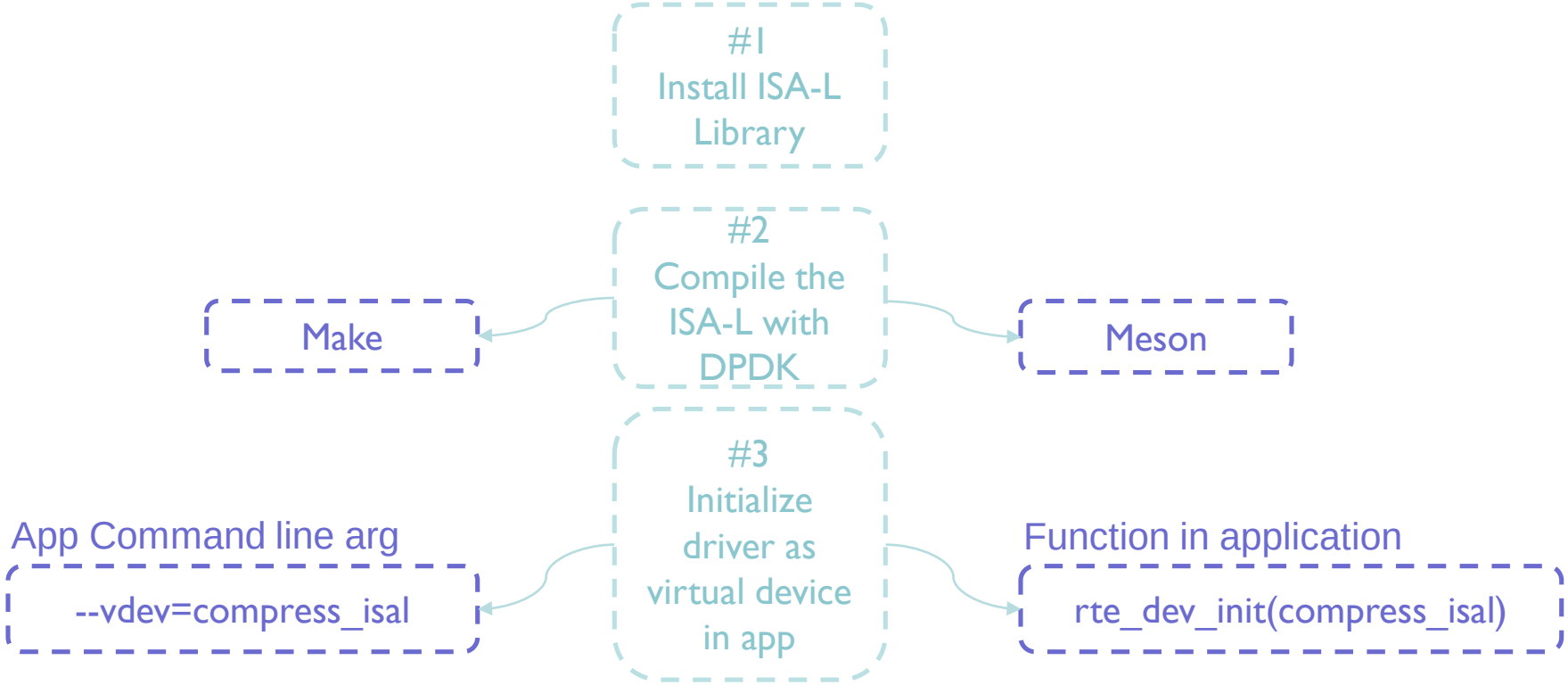
Allows for Shareable Private Xfrom

Single & Chained Mbuf Support

Virtual Device [--vdev=]

Fixed & Dynamic Huffman

How to use the ISA-L PMD



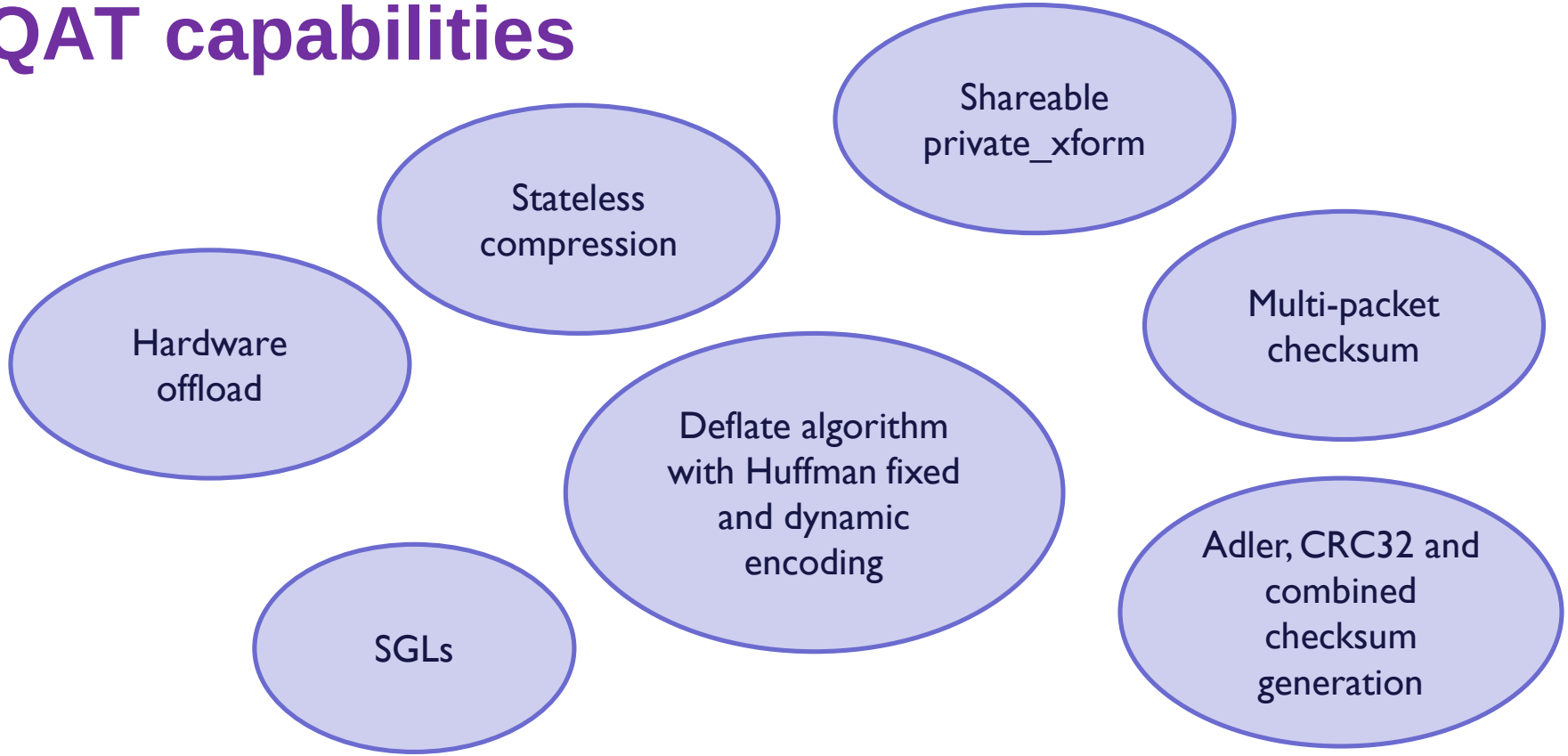
Agenda

- Overview of DPDK compressdev
- Deep dive into some API concepts
- Poll-mode drivers
  - Intel QuickAssist PMD
  - Intel ISA-L PMD

Intel QuickAssist (QAT) poll mode driver

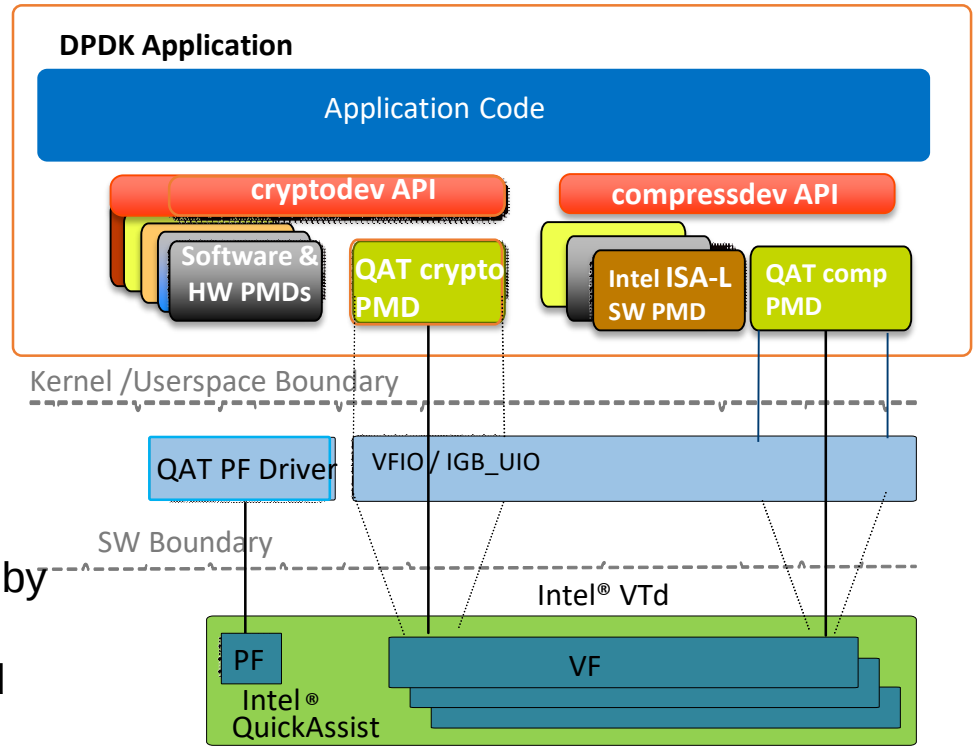
- Offloads compression operations to QAT hardware engines
- Many engines so can process ops in parallel
- MMIOs are expensive so offload cost can be minimised by sending larger bursts

QAT capabilities



QAT PMD depends on QAT kernel driver

- Application Code
- DPDK API
- DPDK QAT Userspace PMD
- Kernel Device Driver
- Physical Device



- QAT kernel driver initialises PF device and exposes Virtual Functions
- VFs are enabled to DPDK QAT PMDs by vfio-pci or igb_uio
- A QAT VF can support both crypto and compression PMDs simultaneously

Questions?

More information @

- http://doc.dpdk.org/guides/prog_guide/compressdev.html
- http://doc.dpdk.org/guides/compressdevs/index.html

or contact us @

- fiona.trahe@intel.com
- lee.daly@intel.com