Compression support in OpenDataPlane (ODP)

Narayana Prasad Athreya

Linaro Connect BUD17
Background

• What is Compression?
  • Reduces bits needed for data representation.
  • It is used widely in networking and storage.
  • Common Compression Algorithms - DEFLATE, LZO

• What is OpenDataPlane?
  • Networking data plane API specification.
  • Allows portable data plane applications to leverage vendor specific hardware.
Compression API

Two schemes are possible:

- Reuse the existing Cryptography API to support Compression
- Introduce independent Compression API as a separate module
Reuse existing API

• Why leverage Cryptography API?
  • Compression is bundled with Encryption as a transformation protocol.
  • Crypto API provisions align with Compression needs
    • Asynchronous mode
    • Notion of sessions
    • Segmented input/output handling
Reuse existing API

- **Pros**
  - Cryptography and Compression in a single operation
  - Fewer API for application to deal with and makes the API look flexible
  - Useful for Hardware that supports both in single operation

- **Cons**
  - Many permutations of operations makes implementation complex.
    - Data ranges for each of Compression, Authentication Cipher may be different.
  - Order of operations
  - Types of output memory - in-place, allocate vs app provided buffer.
  - Not every permutation maybe supported by hardware.
  - Lack of support for Stateful Compression.
Compression API module

- Proposed Compression API usage.
- Borrows ideas from Cryptography API
- Adds notion of stream for Stateful Compression
- Adds compression-specific provisions
  - History
  - Checksum/Digest
Compression API module

- Example usage with multiple chunks

```c
// Application Init-time action
odp_comp_capability();
odp_comp_alg Capability(ODP_COMP_ALG_DEFLATE, alg_cap, num);

// Application Control plane action
sess_params.comp_alg = ODP_COMP_ALG_DEFLATE;
 sess_params.op = compress ? ODP_COMP_OP_COMPRESS : ODP_CRYPTO_OP_DECOMPRESS;
 odp_comp_session_create(params, session, &status);   

// Application Data plane action

/**
 * op_params.out_pkt will hold the output text
 * op_params.comp_alg_op_alg deflate dict will hold history
 */

/**
 * First Chunk (maybe from a file/stream) in op_params.pkt */

odp_comp_operation(&op_params, &result); 

/**
 * Second Chunk in op_params.pkt */

odp_comp_operation( &op_params, &result); 

/**
 * Wait till operation completes and take appropriate action on partial output
 */

/**
 * Last Chunk in op_params.pkt */

op_params.last = TRUE;

odp_comp_operation(&op_params, &result); 

// Wait till operation completes and take appropriate action on partial output

odp_packet_free(op_params.pkt);

odp_packet_free(op_params.out_pkt);

// Application Control plane action

odp_comp_session_destroy(session);
```
Compression API module

**Pros**
- Geared towards Compression
- No modifications to existing ODP API specification
- Implementations could be faster for Compression-only use-cases.

**Cons**
- Requires 2 trips to hardware if Cryptography is also used.
Status

- Reuse Cryptography API
  - Implemented Compression API for linux-generic
  - Work in progress on OCTEON TX ZIP offload Engine.

- Compression API module
  - Work in progress on Compression API module for linux-generic and OCTEON TX ZIP offload Engine.

- Roadmap
  - Run performance benchmarks and compare ODP vs non-ODP.
  - Add IPComp stack and enable compression in ODP.
Q & A

Narayana Prasad Athreya
Prasad.Athreya@Cavium.com