DPDK
DATA PLANE DEVELOPMENT KIT

DPDK Regex subsystem

ALEX ROSENBAUM, MELLANOX

Agenda

- Regex use cases
- Regex device high level APIs
- Future extensions for inline acceleration
- Q & A

Regex DPDK's RFC

- Targeted to standardize the RegEx/DPI offload APIs for DPDK

- The RFC crafted based on SW Regex API frameworks such as:
  - **Perl Compatible Regular Expressions** (libpcre)
  - Intel's Hyperscan

- The API schematics are based ethdev, cryptodev, and eventdev existing device API.

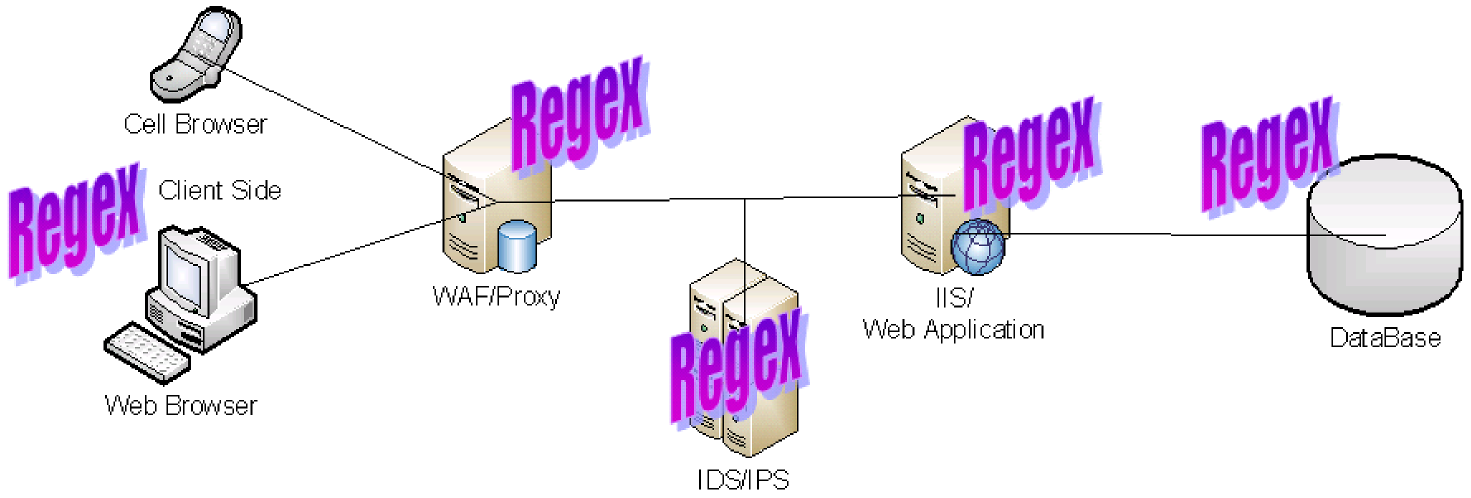
- RFC on mailing list:

```
[dpdk-dev]  [RFC PATCH v1] regexdev: introduce regexdev subsystem
```

https://inbox.dpdk.org/dev/20190627155036.56940-1-jerinj@marvell.com/

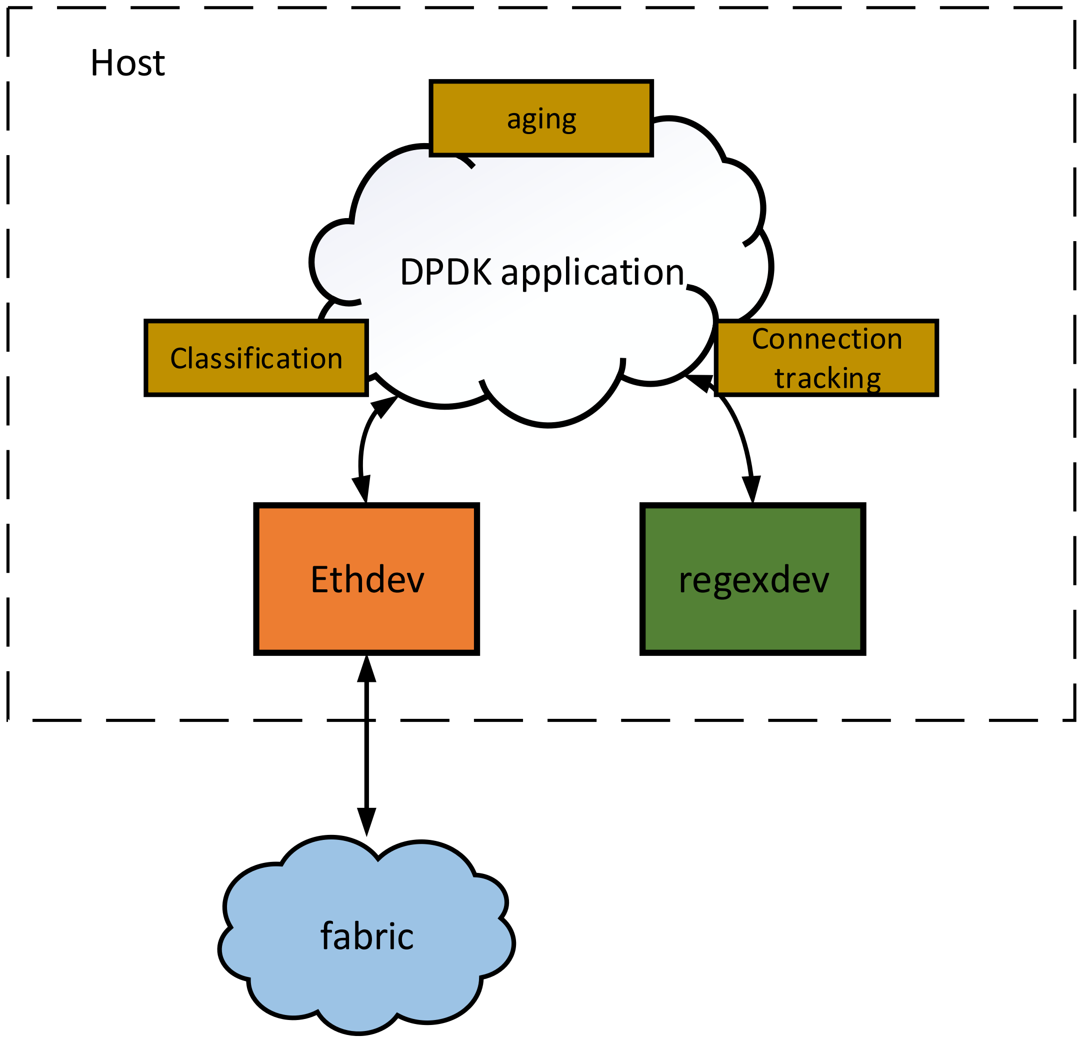
Main use case we target

- Application Recognition
- Intrusion Prevention (IPS\IDS)
- Next Generation Firewall (NGFW)
- Web Application Firewall (WAF)
- Host memory introspection
- DDoS Mitigation
- Network Monitoring
- Data Loss Prevention
- Financial data mining
- Natural Language Processing (NLP)
- Sentiment Analysis



Regex device as look aside accel

- DPDK application do
  - Orchestration
  - Flow Classification
  - Connection Tracking
  - Aging
- DPI library
  - Performs L7 parsing
  - Cross packet matching
- Regex device performs pattern matching:
  - HW device matches 10000’s of compiled signatures in a single path



Few RegEx definitions

- **RegEx**: A regular expression is a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters. A common abbreviation for this is “RegEx”.

- **RegEx device**: A hardware or software-based implementation of RegEx device API for PCRE based pattern matching syntax and semantics.

- PCRE RegEx syntax and semantics specification:
  - http://regexkit.sourceforge.net/Documentation/pcre/pcrepattern.html

Few RegEx definitions

- **Rule**: A pattern matching rule expressed in PCRE RegEx syntax along with Match ID and Group ID to identify the rule upon the match
- **Rule database**: The RegEx device accepts regular expressions and converts them into a compiled rule database that can then be used to scan data
  - Allows analyze of the a given pattern(s) in an optimized fashion
- **Rule ID**: A unique identifier provided at the time of rule creation for the application to identify the rule upon match
- **Group ID**: Group of rules can be grouped under one group ID to enable rule isolation and effective pattern matching
- **Scan**: A pattern matching request action through **enqueue** API

```
  Rule | Database
+------+-----------+
|     Group 0      |
| +--------------+ |
| | Rules 0..k   | |
| +--------------+ |
|     Group 1      |
| +--------------+ |
| | Rules 0..l   | |
| +--------------+ |
|     Group 2      |
| +--------------+ |
| | Rules 0..m   | |
| +--------------+ |
|     Group n      |
| +--------------+ |
| | Rules 0..n   | |
| +--------------+ |
+------------------+
```

DPDK RegEx devices

- Registration:
  - Dynamically registered during the PCI/SoC device probing phase performed at EAL initialization time
  - regex_dev_init():
    - ➢ resetting the hardware or software RegEx driver implementations
    - ➢ registers a ‘`struct rte_regex_dev`‘

- Capabilities:
  - The application may probe unsupported RegEx device features through struct ‘`rte_regex_dev_info::pcre_unsup_flags`’
    NOTE: each regex device might support a different set PCRE features

DPDK RegEx devices API

- Setup call order:

```
rte_regex_dev_configure()
rte_regex_queue_pair_setup()
rte_regex_rule_db_update()
rte_regex_dev_start()
```

- Fast path

```
rte_regex_enqueue_burst()
rte_regex_dequeue_burst()
```

- Teardown:

```
rte_regex_dev_stop()
rte_regex_dev_close()
```

Managing the Rule Database

-
  ```
  enum rte_regex_rule_op {
   RTE_REGEX_RULE_OP_ADD,      /**< Add RegEx rule to rule database */
   RTE_REGEX_RULE_OP_REMOVE    /**< Remove RegEx rule from rule database */
  };
  ```

-
  ```
  struct rte_regex_rule {
   enum rte_regex_rule_op op; /**< OP type of the rule either a OP_ADD or OP_DELETE */
   uint16_t group_id;         /**< Group identifier to which the rule belongs to. */
   uint32_t rule_id;          /**< Rule identifier which is returned on successful match. */
   const char *pcre_rule;     /**< Buffer to hold the PCRE rule. */
   uint16_t pcre_rule_len;    /**< Length of the PCRE rule*/
   uint64_t rule_flags;       /** @See RTE_REGEX_PCRE_RULE_* */
  };
  ```
- `rte_regex_rule_db_update() /* runtime compile of the PCRE rule database */`
- `rte_regex_rule_db_export() /* device specific buffer of pre-complied rules */`
  `rte_regex_rule_db_import()`

more

- Additional regex device attribute set/get
  - rte_regex_dev_attr_set()
  - rte_regex_dev_attr_get()

- Stats and debug
  - xstats:
    - ➢ rte_regex_dev_xstats_get()
    - ➢ rte_regex_dev_xstats_names_get()
    - ➢ rte_regex_dev_xstats_by_name_get()
    - ➢ rte_regex_dev_xstats_reset()
  - rte_regex_dev_dump()
  - rte_regex_dev_selftest()

Where we are heading

- Connection Awareness subsystem
  - Bi directional connection awareness
  - TCP connection tracking
  - Aging mechanism
  - Flow ordering and re-assembly.

- DPI library
  - Cross buffer Regex inspection
  - L7 protocol parsing

- Security offload demo DPDK application
  - Inline acceleration by HW of crypto, DPI and connection awareness ahead of the host (e.g. SoC)

Thanks !

Jerin Jacob jerinj@marvell.com
Shahaf Shuler shahafs@mellanox.com