DPDK

DATA PLANE DEVELOPMENT KIT

Do Less By Default

BRUCE RICHARDSON - INTEL

THOMAS MONJALON - MELLANOX

Introduction

WHY?

Consumability

Make it easier to:

- take bits (slices) of DPDK
- fit DPDK into an existing codebase
- integrate existing functionality into a DPDK app

Don’t Offer Less!

- Key phrase “***by default***”
- Provide array of re-usable components
- Make it trivial to do things the default way
- **Aim:**
  - ensure external tools have a path to work with the majority of DPDK apps!

By Matt @ PEK from Taipei, Taiwan - Buffet breakfast, CC BY-SA 2.0

Episode I

IN WHICH OUR HEROES
EXAMINE THEIR OPTIONS

Configuration Options Issues

- Command line options parsing done by DPDK EAL
  from arguments passed to `rte_eal_init(int argc, char **argv)`
- Hard to translate settings from the application to this syntax
- Some configuration cannot be changed later with simple API function call
- One benefit: applications are encouraged to use the same syntax

Suggestion: New Option Store Library

- Functions to parse all as in legacy rte_eal_init

```
rte_opt_parse_argv(int argc, char **argv)
rte_opt_parse_args(const char *args)
```

- More fine grain parsing

```
rte_opt_parse_kv(const char *key, const char *value)
```

- Parsed values are written into a big structure `rte_opt_settings` for all

- DPDK libraries should not read settings directly from the structure

Suggestion: Options Store for DPDK Init

- Leverage new library to parse options with default syntax
  - Keep same syntax or maintain compatibility
- Application is free to use the default parser or not

- New wrapper function, calling initialization functions with parsed settings or default values

```
rte_default_init()
```

- Then deprecate `rte_eal_init()` ?

Future Considerations

- The new devargs syntax can be used in bus, device or driver settings
- Build-time settings should be almost all replaced by run-time options

Episode II

IN WHICH OUR HEROES DEAL
WITH SOME CORE ISSUES

Core Management Issues

Public Domain, Link

- EAL wants to do all core and thread management
- DPDK requires a coremask for EAL init
- If no coremask given, spawns thread for every core on system!
- Even for spawning no threads, still affinitizes current thread to a core
- ***How do you integrate DPDK into an existing multi-threaded app?***

Suggested Changes

- Allow “-c 0” as coremask – do nothing!
    - Don’t spawn any worker threads
    - Don’t set affinity of master (current) thread

- Change behaviour for empty core mask – do nothing!

- Add API’s for explicit thread management by app, e.g.:
    - `rte_thread_init()` – allocate lcore_id, FIFOs etc.
    - `rte_thread_process()` – accept DPDK work via FIFO, as per existing threads
    - `rte_thread_process_one()` – accept one job from DPDK, then return to caller
    - `rte_thread_cleanup()`

Future Considerations

- How to allow orchestration of DPDK apps?
- How to enable app scale-up and scale-down?
- Needs common/default orchestrator-to-app comms
- Then needs some form of callback mechanism in app
- Built into EAL, ***BUT:***
  - needs to keep app in control!
  - needs to be optional feature!

DPDK
DATA PLANE DEVELOPMENT KIT

Episode III

IN WHICH OUR HEROES GET CONSTRUCTIVE

Constructors Issue

`__attribute__((constructor))`

DPDK cannot be fully disabled – Constructors are **always enabled**

- Functions declared with `RTE_INIT()` macro
  run before `main()`
  even if DPDK not initialized
- Application packaged with DPDK
  may disable DPDK acceleration at run-time
  if hardware not supported
- On x86, DPDK is compiled for SSE4.2 minimum
- Crash happens in useless DPDK constructor **if CPU is too old**

Suggested Changes

- Add `__attribute__((target(minimum)))` to `RTE_INIT()`
- The minimum can be `default`, `sse2`, etc

- Option 1
  - Must apply target restriction to all functions called in constructors
  - Hard to maintain

- Option 2
  - Insert call to `rte_cpu_is_supported()` in `RTE_INIT()`
  - Apply target restriction to CPU check functions
  - Skip constructor code if CPU is not supported

Future Considerations

- Is it sane to keep using constructors in a library like DPDK?
- Could be changed in simple functions
  called at the beginning of the DPDK initialization?

DPDK
DATA PLANE DEVELOPMENT KIT

Episode V

IN WHICH OUR HEROES END WITH AN OFF-BY-ONE ERROR