BROCADE

ubuntu

deb_dpdk

Challenges and Opportunities when packaging DPDK

DPDK Summit Userspace Dublin - 2016

Linaro

CANONICAL

Introduction

Luca

Working for Brocade as member of the Vyatta Virtual Router team

Packaging DPDK in Debian

Debian Maintainer

ZeroMQ developer and maintainer

Christian

Working for Canonical as member of the Ubuntu Server Team

Packaging DPDK in Ubuntu

Also working on qemu, libvirt, ntp, cloud-init, curtin and actually any package of the server seed as needed.

Past life includes KVM development as well as Linux & KVM performance on Mainframes.

Background

- Brocade packaged dpdk 1.6, 1.8, 2.0 and 2.2 internally for their Debian based Product, moving to 16.07 at the moment
- Christian packaged dpdk 2.0, 2.2 and 16.07 for Ubuntu
- Since 16.04 we work together at https://wiki.fd.io/view/Deb_dpdk to bring it to Debian and Ubuntu in a unified way
- There were some hurdles, most of the time due to DPDK behaving different than most other libraries. Looking back all bad mood is gone and we call them lessons learned.

Challenges

Library versioning

- <=2.2 it was hard to assign a soname at all
  - Only after that libdpdk.so as linker script somewhat solved the case
    - causes dpdk to be split into many packages for individual dependencies in .deb world
  - The ABI will continue to be a problem
    - Each ABI bump means a multi-week transition
    - Dependency Deadlock - All depending “In-archive” packages have to be rebuilt
      - To be able to do so they have to be compatible and not many match your pace
      - Currently only Open vSwitch, but we see OpenDataPlane and VPP on the horizon
    - Out of archive solution have to rebuilt on their own on each bump
  - Yet unclear what will happen if we end up with mixed dependencies
    - In less split libraries packagers can provide both when libfoo2 gets bumped to libfoo3
    - But a dpdk abi bump affects only some libs, those rte libs without a bump will conflict

Runtime environment

- Users usually expect configuration files
  - attaching cards / assign drivers
  - reserving hugepages
  - EAL arguments (socket, bitmask, memory, numa nodes, …)
- The expect things to be stable across reboots and that basics like hugepage mount points are taken care of.
- Config files are expected to be stable across updates (Maintainer scripts)
- DPDK is great, but those bits are what makes it easy to consume

Further packaging hurdles and features

- Why does one have to build twice for static + shared libraries?
- Custom rebuilds with custom DPDK_CONFIG, RTE_MACHINE, RTE_TARGET
- Kernel modules for kni and igb_uio
  - Fortunately dkms covers distributing in a way that it is buildable for local kernels
  - Pre-built binary kernel modules packages are provided optionally
- Many severe bugs were still found on a almost daily base for a while. Things settled down a bit, but there still is a high backport & maintenance effort.
- Old vs New DPDK philosophy
  - Old: build an app statically linked with DPDK - use cases and paths clearly defined
  - New: dpdk as a normal library - all paths possible, much harder to optimize

Opportunities

Optimization I

- Path optimization
  - Different preferences depending on use cases
  - Pluggability vs speed
  - While single projects exploiting DPDK can decide their preferences a generic library can’t
    - For example VPP has different optimization wishes for DPDK than others
    - An appliance based on DPDK can take focus on just what it does and the HW it will be deployed on
    - but what do we do?

Optimization II

- CPU optimization
  - Distributions (have to) aim for a high compatibility to various HW
    - Debian/Ubuntu amd64 targets all x86_64 CPUs, not just Sandy Bridge+ !
  - Use runtime optimization over buildtime optimization whenever possible
    - discussed multiple builds, switched via update alternatives, but it is a maintenance nightmare
    - wherever else that was needed it was solved via runtime detection

Availability of dependencies

- A distribution not only aims for high HW but also SW compatibility
  - Example: The mlx pmd drivers are only compatible with the vendor specific MLNX_OFED
  - Hard for Distributions
    - Used to handle backports/forward-ports
    - But becoming vendor specific usually is a no go
  - To some extent that is more an openfabrics than a DPDK issue
  - But dpdk should try to build against open libibverbs-dev packages

Stable Trees

We welcome the founding of a stable tree (again)

- That is especially important given the effort of an ABI bump transition
- Eases maintenance for the many downstreams
  - Important for bug and security fixes
- So far discussion and adoption rate is low
  - Recently stable was adopted, so looking forward to see activity once 16.11 is finalized
- “Backward maintainability” also is something to consider
  - I totally understand and don’t object commits like:
    “This patch implements the [...] logic from scratch into a single function to improve maintainability”
  - but you can also translate that with “make it impossible to backport later fixes”

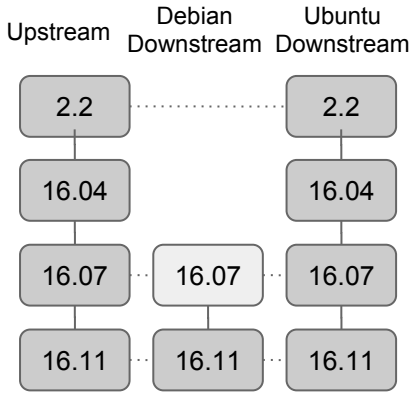
Stable Trees

Stable distributions (Debian, Ubuntu, enterprise distros)

- golden rule: better the bugs we know than the bugs we don’t know
    - Once Debian/Ubuntu stable ships, CANNOT break ABI ever again
    - No new versions uploads as a general rule, few exceptions
    - Only security/critical fixes allowed as out-of-tree patches
- Upstream stable tree fundamental to cherry-pick fixes from for 2/4 years
- 16.07 in Ubuntu 16.10, aiming for 16.11 in Debian 9

Runtime unification

- We want to feed back our packaging into dpdk.org as soon as we have stabilized it enough
- That allows to reuse and jointly improve e.g. the runtime bits like init scripts, config structure and such even across distributions
- it allows to align changes in source with changes to runtime bits
- dep_dpdk will stay even after upstreaming those
  - As our boilerplate for any new deb packaging feature
  - Place to maintain certain distribution release branches
    - Debian Unstable rolling distro, migrate to testing, freeze to stable
    - Ubuntu derives from Debian without (sync) or with delta (merge)
    - Each Release is serviced for some time
    - deb_dpdk is where we coordinate and plan that to help instead of handicap each other



Thank you

- We are grateful for the great collaboration in the dpdk project whenever we brought something up
- As you have seen the .deb packaging itself is a collaboration project
- We look forward to tackle some of the opportunities together
- If anybody wants to participate take a look at deb_dpdk
  - take a look how we handle runtime things
  - contribute if you have needs that should be solved at the packaging level
  - Discuss
- Many thanks also to Ed Warnicke, C.J. Collier, Ricardo Salveti de Araujo and Santiago Ruano Rincón for all their help on packaging and infrastructure!

DPDK

Questions?

Open discussion on:

- Usability of DPDK
- Packaging
- Stable releases / LTS
- ...