Reducing Barriers to Adoption
Making DPDK Easier to Integrate into Your Application

Bruce Richardson
DPDK Summit - San Jose – 2017

#DPDKSummit
Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, and other Intel product and solution names in this presentation are trademarks of Intel . . .

*Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.
How “consumable” is DPDK?

Chocolate Fudge Cake by Tracy Hunter is licensed under CC BY 2.0

© User:Colin / Wikimedia Commons / CC BY-SA 3.0

#DPDKSummit
DPDK as Broccoli

- Not the most consumable thing in the world
- But it’s fine when you get used to it!

- That does not mean we shouldn’t strive to achieve “DPDK as Cake” status
DPDK is Greedy

- DPDK thinks it owns all cores and threads in the app
- DPDK thinks all hugepage memory on the system is for its exclusive use
- DPDK thinks it should own the application command-line
- DPDK thinks you want to build all DPDK apps using the DPDK build system
DPDK is Greedy

- DPDK thinks it owns all cores and threads in the app
- DPDK thinks all hugepage memory on the system is for its exclusive use
- DPDK thinks it should own the application command-line
- DPDK thinks you want to build all DPDK apps using the DPDK build system
Reduce the dependency on physically contiguous memory
  - Few cases actually need this
  - Eliminates the need to find biggest contiguous blocks of memory

Allow memory/hugepage allocation from OS on demand
  - Faster startup
  - Memory footprint is only what the application needs

Allow free memory to be given back to the OS
Memory – Work In Progress

- Requires very considerable rework on DPDK internals
  - Hugepage handling
  - Memory segments (memsegs)
  - Memory zones (memzones)
  - Malloc (rte_malloc)
  - + changes to other components needing contiguous memory

- Patches expected in 18.02 timeframe

- No expected performance implications!
Still not the most consumable thing in the world

Many folks ok with it as-is

But with a bit of work it can go into many very consumable things!
DPDK build system is currently based on “make” alone
- Static configuration only - no dynamic configuration at build time
- Custom build system with many complicated makefiles
- Assumes that it will be used to compile end-user apps too

Work underway to replace this with new build using “meson”
- Very popular with open-source projects, e.g. Xorg, system
- Dynamic build configuration with a readable syntax

Likely to be a multi-release effort
- Old build system is not going away any time soon!
pkg-config – Improved Consumability

- pkg-config
  - Standard method to specify how to compile and link with a library
  - Very distro-friendly

- No need for either:
  - trying to hack some/all of DPDK build system into your projects
  - hard-coding the libraries & ldflags into your makefiles

PC_FILE := $(shell pkg-config --path libdpdk)
CFLAGS += $(shell pkg-config --cflags libdpdk)
LDFLAGS += $(shell pkg-config --libs libdpdk)

helloworld: main.c Makefile $(PC_FILE)
  $(CC) $(CFLAGS) main.c -o $@ $(LDFLAGS)
bruce@host:dpdk-next-build$ grep Xeon /proc/cpuinfo | head -1
model name : Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz

bruce@host:dpdk-next-build$ grep Xeon /proc/cpuinfo | wc -l
72

bruce@host:dpdk-next-build$ make config T=x86_64-native-linuxapp-gcc
Using DPDK – a piece of cake?

- Two changes will not fix all problems
- But fixing these two should move us considerably along the consumability path
- Suggestions, and patches very welcome for other future changes

Who wants DPDK improved? – a pie chart

By Oddbodz - Own work, CC BY-SA 3.0, Link
Questions?

Bruce Richardson
bruce.richardson@intel.com