Packaging Tensorflow 2.5.x

for fun and profit

DebianReunionHamburg - 2022

Felix Mößbauer, Jan Kiszka

SIEMENS

Introduction

SIEMENS

About me

Scientific Background

Master Computer Science @ LMU (2019)
Software Engineer @Multicore Expert Center, Siemens

Focus on:

- High Performance Computing (@see DASH-Project)
- Efficient implementation of algorithms (C/C++/ASM)
- Low latency & realtime communication

Influxdb-cxx DASH DRace DPDK rtl_433

Roslyn2Famix ***Open Source Enthusiast*** HPCCG

Mosquitto SoapySDR ShinySDR Prometheus

Find me on

github.com/fmoessbauer
www.linkedin.com/in/felixmoessbauer

Felix Mößbauer

SIEMENS

Background

We have…

- **devices**: SIMATIC IOT2050 (arm64), IPCs (x86_64)
- ISAR / Debian Board Support Package
- based on Debian bookworm

Google Coral EdgeTPU (coral.io)

SIMATIC IOT2050 – ARM Cortex A53

Our Task

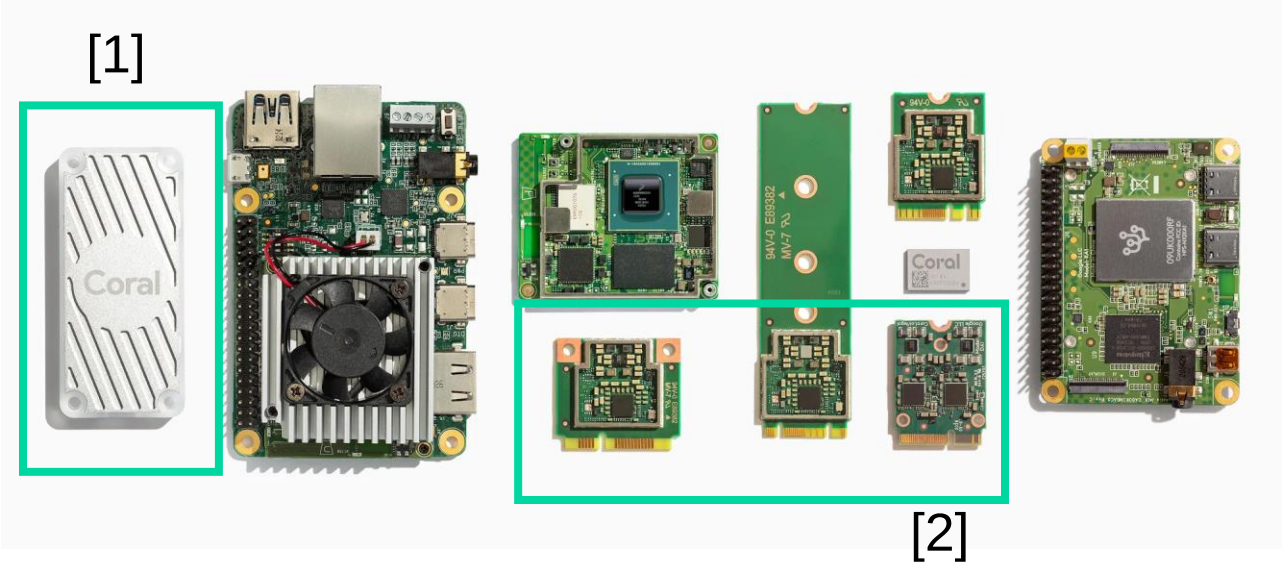
- integrate support for Google Coral EdgeTPU chip
- In a way that is not too fuzzy
- fully cross-build compatible

SIEMENS

Background

Things to consider…

- SBOM
- Close to upstream
- Maintainable



Google Coral EdgeTPU (coral.io)

No Gos

- Use pre-built packages from Google
- Build without Debian standard tools (e.g. *dpkg-buildpackage*)
- Fetch dependencies at build-time

[1]: USB Coral

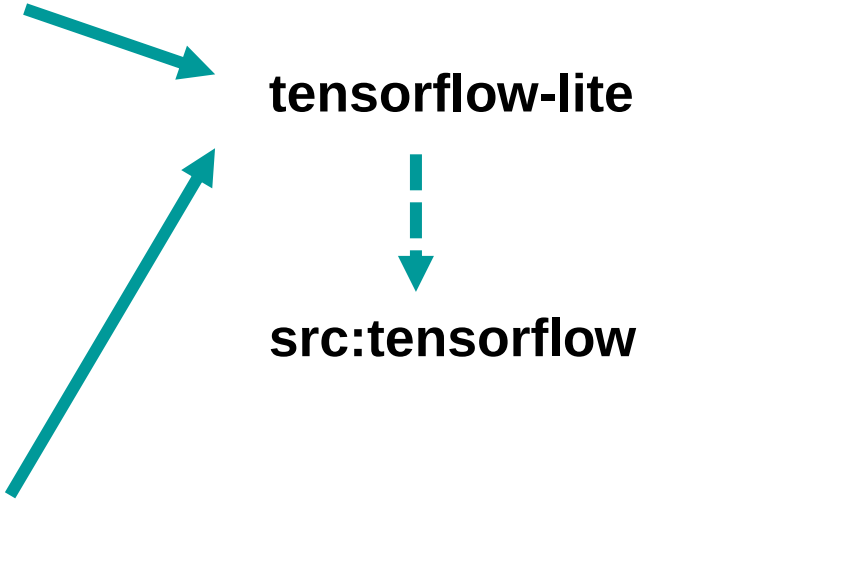
[2]: PCIe / M2: Kernel module required

SIEMENS

Dependencies

Buildtime and Runtime deps of Google Coral

package	amd64	arm64	notes
~~edgetpu-compiler~~	~~N~~	~~N~~	~~binary-only~~
libedgetpu1-max	T	T	
libedgetpu1-std	T	T	
libedgetpu-dev	T	T	
gasket-dkms	T	T	
gasket-module	T	T	
python3-pycoral	T	T	



N = not applicable, T = currently being tested

SIEMENS

Dependencies – How are they handled by bazel

The bazel build-system

- incompatible with the rest of the world
- all deps are handled internally (via source-code fetchers)
- Bazel versions: not too new, not too old (ref basilisk)

System Dependency Injection

- support for system dependencies has to be explicitly handled by bazel rules of project (TF_SYSTEM_LIBS)
- cross-compilation: *oh dear...*
- caching: always in $HOME

Dependencies – Pre-Fetch all non-Debian deps

```
237  .PHONY: local
238  local:
239          $(prepare_bazel_env)
240          bazel sync $(foreach r,$(EXTERNAL_SOURCES),--only=$(r))
241          mkdir -p $(@)
242          $(foreach r,$(EXTERNAL_SOURCES),                                    \
243                  echo "# importing $(r) from bazel cache to $(@)/..." &&    \
244                  mv $(HOME)/.cache/bazel/_bazel_root/*/external/$(r) $(@)/ && )    \
245          symlinks -cr .
246          $(clean_bazel_env)
```

We relocate the components, but they have inter-component symlinks

Dependencies – Prohibit download at build-time

```
184  # use invalid proxies to make sure we do not need to fetch anything
185  set_invalid_proxies =                                                \
186          export ftp_proxy=http://127.0.0.1:1;                         \
187          export http_proxy=http://127.0.0.1:1;                        \
188          export https_proxy=http://127.0.0.1:1;

125  export CUSTOM_BAZEL_FLAGS =                                                  \
126                  --repo_env=TF_SYSTEM_LIBS=$(system_libs)                     \
127                  $(foreach r,$(EXTERNAL_SOURCES),--override_repository=$(r)=$(CURDIR)/local/$(r))
128
```


SIEMENS

Cross-Compiling & Maintenance

SIEMENS

Build times are extreme – We need cross-compilation support

TF Debian Dependencies

- Many python packages
- Non-pure python packages often have cross-compiling issues

```
-           python3-all,
+           libpython3-all-dev,
+           python3-all-dev:any,
```

Cross-compiling with bazel

- Toolchain file required
- Toolchain has to be patched into project
- ~1000 LOC to describe a toolchain (CMake: <10)

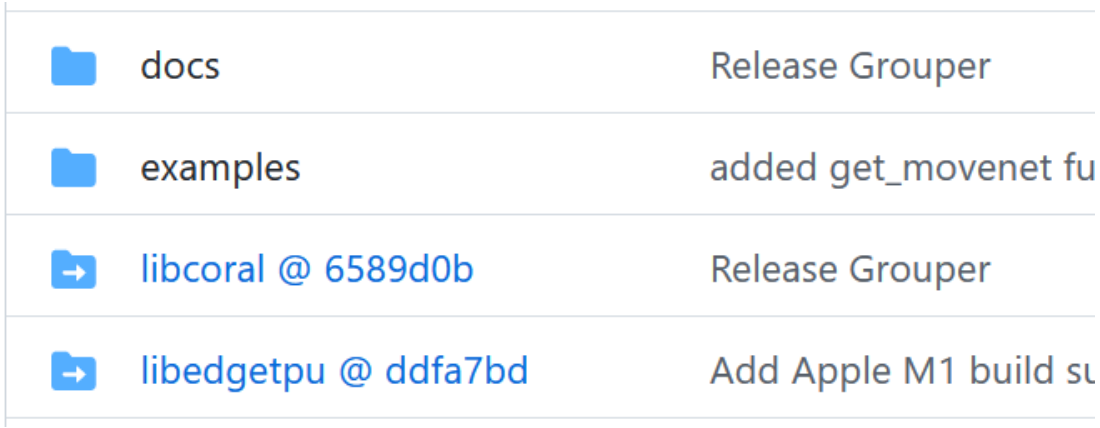
```
else ifeq ($(DEB_HOST_ARCH),armhf)
        CUSTOM_BAZEL_FLAGS += --cpu=armhf --crosstool_top=@system_config_arm_compiler//:toolchain
else ifeq ($(DEB_HOST_ARCH),arm64)
        CUSTOM_BAZEL_FLAGS += --cpu=aarch64 --crosstool_top=@system_config_arm_compiler//:toolchain
endif
```


SIEMENS

Maintenance – Version bumps

The path we travelled

- 2.3.x -> 2.5.1
- 2.5.1 -> 2.5.3



https://github.com/google-coral/pycoral

Lessons learned

- Even minor version bumps of TF change things all over the place
- Significant effort required for re-packaging and testing
- Due to nature of bazel, inter project dependencies are layed out on exact git SHAs (e.g. via git-submodule) or versions (via fetchers)

SIEMENS

Outlook

SIEMENS

Make upstreaming as easy as possible

Limitations

- here and there we violate Debian policies
- still not sbuild compatible
  (we are working on it)
- we didn't put too much effort into packaging the build tooling (bazel, toolchain, etc...)
- Maintenance unclear

TODOs

- Discuss topics with TF and Google (again)
- have more recent versions of toolchain in Debian
- Add Debian tooling around bazel to simplify dependency handling

Contact

Felix Mößbauer
felix.moessbauer@siemens.com

Jan Kiszka
jan.kiszka@siemens.com


github.com/siemens/meta-coral


github.com/siemens/meta-iot2050

Image Sources:

IOT2050: https://new.siemens.com/de/de/produkte/automatisierung/pc-based/iot-gateways/simatic-iot2050.html

Coral: https://coral.ai/products/

SIEMENS