TF-M split build

continue

Anton Komlev
September 2023
Background and problem definition

- A single build process for 3 binaries
- Sharing config options leads to:
  - Large and complex configuration set
  - High entrance barrier for TF-M App developer
  - Maintenance difficulty
  - Error-prone and vulnerable to side-effects
- Require tricks in CMake build script to support different CPUs on S and NS
- Build starts from SPE - reverse logic
- Can we reduce dependencies in this Client - Server scenario?
  - BL1, BL2 and S configurations are mainly defined by HW platform
  - Separate codebase for NS and S sides
Split build alternative
2 semi-independent projects

TF-M = SPE = Secure(S) side

+ Developer selects
  - A platform
  - Secure service set
+ A platform has highest config priority
  - CPU cores and HW capabilities
  - Memory layout and peripherals
+ Outputs = exports = installs
  - PSA interface
  - BLs, S binaries
  - Bin image tools (signing, merging)
  - NS toolchains
  - NS platform
    - Sources
    - MCPU + Arch

Application = Non-Secure(NS) side

+ An application code
+ Builds and links with NS platform sources
+ Combines with BL, S binaries
+ BLs and S
  - Stay the same
  - OEM can ship it in binaries
+ Is independent from S build
  - Toolchain and options
  - S source tree
Implementation

- Installation script is extended to install:
  + Common NS platform files
    + CMSIS
    + Toolchain
    + Link scripts/scatters
    + CMakeLists.txt for platform_ns
  + spe_config.cmake
  + spe_export.cmake
  + CMakeLists.txt for SPE
- A Platform installs
  + NS platform sources
    + Startup
    + Drivers
  + Linker script (scatter)
- TF-M does not dictate how to build TF-M application
Configuration

Most of configuration options are on S side

S side

+ Configurations
  - Platform selection
  - Secure services
+ Config mechanisms
  - Kconfig
  - Predefined or custom profiles
  - CLI settings
+ A small subset of config options carried to NS application because:
  - A platform is selected
  - Partitions are defined

NS side

+ Nothing to configure for TF-M itself
+ App can retrieve some SPE options

```c
#include(spe_config)
#include(spe_export)

add_library(tfm_api_ns)
add_subdirectory(platform)
target_link_libraries(tfm_api_ns
  platform ns
tfm_config)
```

```
spe_config
spe_export
platform/CMakeLists
CMakeLists
- variables
- definitions
```
Platform porting steps

- Move S side **CPU** and **Arch** definitions from preload.cmake → config.cmake
- Add installation instructions for NS platform sources to CMakeListst.txt.
  - Following destination variables are available:
    - INSTALL_INTERFACE_INC_DIR - <dst>/interface/include
    - INSTALL_INTERFACE_SRC_DIR - <dst>/interface/src
    - INSTALL_INTERFACE_LIB_DIR - <dst>/interface/lib
    - INSTALL_IMAGE_SIGNING_DIR - <dst>/image_signing
    - INSTALL_CMAKE_DIR - <dst>/cmake
    - INSTALL_PLATFORM_NS_DIR - <dst>/platform
- All files from /ns/ folder will be installed. Those 2 are expected
  - ns/CMakeLists.txt - Script for building platform_ns target
  - ns/cpuarch.cmake - definitions of CPU and Arch
- Remove
  - preload.cmake
  - Traces of platform_ns, NS from <tf-m platform>/CMakeListst.txt
- Musca-B1 porting example:
  [https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/23468/](https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/23468/)
- Takes about 1 day (with a luck)
“Hello TF-M” demo app

Ref: tf-m-extras/tf-m-example-ns-app  [not merged at the demo time]

Main.c

```c
/* Copyright (c) 2023, Arm Limited. All rights reserved.
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "tfm_api.h"
#include "uart_stdout.h"
#include <stdio.h>

int main(void)
{
    uint32_t fw_version;
    stdin_init();
    printf("Non-Secure system starting...\n\n");
    printf("Hello TF-M\n\n");
    fw_version = psa_framework_version();
    printf("Firmware version = %d.%d\n", fw_version >> 8, fw_version & 0xFF);
    while(1);
}
```

CMakeLists.txt

```cmake
# Copyright (c) 2023, Arm Limited. All rights reserved.
# SPDX-License-Identifier: BSD-3-Clause
#
# cmake_minimum_required(VERSION 3.15)
#
# set(CONFIG_SPE_PATH /home/antkom01/hello-tfm/api_ns)
set(CROSS_COMPILE arm-none-eabi)
set(CMAKE_TOOLCHAIN_FILE ${CONFIG_SPE_PATH}/cmake/toolchain_ns_ARMARM.cmake)
list(APPEND CMAKE_MODULE_PATH ${CONFIG_SPE_PATH}/cmake)
project("TF-M Example") LANGUAGES C)

add_executable(tfm_ns
    ${CONFIG_SPE_PATH}/interface/src/os祉wer/tfm_ns_interlace_bare_metal.c
    main.c)

add_subdirectory(${CONFIG_SPE_PATH} tfm-api-ns)
target_link_libraries(tfm_ns tfm_api-ns)
```
Demo time
Tests

Are TF-M applications. Adopted and decoupled

Key changes

+ Now builds as independent TF-M applications
  - Regression tests
  - PSA-Arch tests
  - ERPC server
+ CONFIG_TFM_TEST_DIR
  - Included as sub_directory() into SPE build
+ NS Tests execution environment is gathered into app_broker target
+ Tests provides only entrance function:
  - void test_app(void *argument)

New structure
Current status

- Ported to an521, Musca-B1, Musca-S1 platforms
- Major regression tests and PSA Arch tests are passed on those platforms
  - Some tests (like FP) are not yet adapted
- OpenCI is ready for basic testing on the staging environment
- Changes are in feature-build-split-v2 branches of TF-M and tf-m-test repositories
  - https://review.trustedfirmware.org/c/TF-M/trusted-firmware-m/+/23572
  - https://review.trustedfirmware.org/c/TF-M/tf-m-tests/+/23209/
- Open technical questions
  - Repositories version’s synchronization: TF-M ⇔ tf-m-tests ⇔ tf-m-extras
    - Released versions are synched by tags
    - Use manual synchronization in a daily work
  - Mechanism for platforms to influence on NS exports
    - extend/redefine/overwrite common NS settings. No need so far but shall be useful in theory
- TODO:
  - Port to remaining platforms
  - Add ArmClang and IAR toolchains support
  - Documentation
  - Port ERPC testing framework
  - Clean Code tree from the single build remains
    - Deprecate obsolete config option
Discussion

Deployment method

1. One time switch in Nov 2023 release → TF-M v2.0.0
2. Offer a deprecation time, but:
   - Overhead in maintaining 2 versions
   - Potential conflicts between them
   - Which version to test in CI
Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
धन्यवाद
תודה