FF-A v1.1 boot protocol implementation (in Hafnium)

João Alves
**FF-A v1.1 Boot Protocol Summary**

- Boot information structures have been updated:
  - Boot information blob = boot info header + boot info descriptors + data.
- FDT boot info type to pass FF-A manifest to the partition.
- Memory management requirement to have the whole boot info blob into a continuous memory region:
  - Simplify memory mapping operation at S1.
Goals

- SPMC interoperability (S-EL1 OPTEE and EL3 SPMC) by adopting the standard defined in FF-A v1.1 EAC0.
- Add ability to pass the SP manifest address using the FDT type.
- Allow for larger SP manifest sizes.
- Add ability to test SP with S1 translation granule larger than 4kb.
Solution Proposal

• Hafnium supports 1-to-1 memory mapping VA to PA.
• To avoid copying FF-A manifest from its loading location, allocate the memory for the boot Information in the SP Pkg:
  o SPTool change (bump package version to 0x2).
  o Hafnium's processing of the SP Pkg.
• Subscription to specific boot information in the SP's manifest.
• Patch stack available in link.
Secure Partition Package Update

- Hafnium supports both versions of the SP Pkg (patch).

**Version 0x1**

- SP Pkg Header
- FF-A manifest
- Image

... 

**Version 0x2**

- SP Pkg Header
- Allocated space
- FF-A manifest
- Image

...
**SPTool update**

- Manifest and Image offset specified through the arguments to the tool
  - Default values: 0x1000 and 0x4000 respectively.
  - Means to the increase the size of the allocated space for the boot info descriptors.
  - Allow for packaging SP's with translation granules of 16Kb or 64Kb.
- Tool updated to python.

```
j oasisv018e124102:/workspace/trusted-firmware-a$ ./tools/sptool/sptool.py --help
Usage: sptool.py [-h] [-i I] [-pm-offset PM_OFFSET] [-img-offset IMG_OFFSET]
  -o O [-v]
optional arguments:
  -h, --help    show this help message and exit
  -i I          path to partition's image and manifest separated by a colon.
  --pm-offset PM_OFFSET set partition manifest offset.
  --img-offset IMG_OFFSET set partition image offset.
  -o 0          set output file path.
  -v            print package information.
```
Hafnium update

- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP Pkg header.
Hafnium update

- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP package header.
  2. Map FF-A Boot Information (RW) and FF-A manifest (RO) sections, into S-EL1 S1 translation.
  4. Write the address of the boot info blob to the register specified in the 'gp-register-num' field.

```c
gp_register_num = <6>;

/* Boot-Info */
boot-info {
  - compatible = "arm,ffa-manifest-boot-info";
  - ffa_manifest;
};
```
Hafnium update

- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP package header.
  2. Map FF-A Boot Info (RW) and FF-A manifest (RO) sections.
  4. Write the address of the boot info blob to the register specified in the 'gp-register-num' field.
- SPMC unmaps from EL-2 Stage-1
SP initialization

- SP boot with S2 translation update.

![Diagram of SP initialization]

```
INFO: Arm SMUv3 initialized
INFO: Loading VM id 0x8001: ffa_secure partition.
VERBOSE: VM has 0 physical interrupts defined in manifest.
VERBOSE: stream_count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x62800000.
INFO: Hafnium initialization completed
VERBOSE: plat_psci_cpu_resume: cpu mpidr 0x0 ON
[fhtest ctrl: get command line]
VM 8001: [fhtest] SP boot info (62800000):
  VM 8001: [fhtest] Signature: ffa
  VM 8001: [fhtest] Version: 10001
  VM 8001: [fhtest] Blob Size: 4033
  VM 8001: [fhtest] Descriptor Size: 32
  VM 8001: [fhtest] Descriptor Count: 1
  VM 8001: [fhtest] Type: 0
  VM 8001: [fhtest] Flags:
    VM 8001: [fhtest] Name Format: 0
    VM 8001: [fhtest] Content Format: 0
  VM 8001: [fhtest] Size: 737
  VM 8001: [fhtest] Value: 6281000
  VM 8001: [fhtest] FINISHED
[fhtest ctrl: finished]
INFO: Loading VM id 0x8001: ffa_secure partition.
VERBOSE: VM has 0 physical interrupts defined in manifest.
VERBOSE: stream_count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x62800000.
INFO: Hafnium initialization completed
VERBOSE: plat_psci_cpu_resume: cpu mpidr 0x0 ON
[fhtest ctrl: get command line]
VM 8001: [fhtest] FF-A Manifest Address: 6281000
VM 8001: [fhtest] FF-A Version: 10001
VM 8001: [fhtest] FINISHED
[fhtest ctrl: finished]
```
TF-A build integration

- Script `sp_mk_generator.py` parses `sp_layout.json` file and generates 'sp_gen.mk'.
- The patch extends the script to generate a make rule for an SP package, for each SP defined in the `sp_layout.json`. 
FF-A manifest configuration

- Adjust the 'entrypoint-offset' to either match the image offset in the SP Pkg.
  - For version 0x2 of the SP Pkg this can be the default value taken by the sptool.py (0x4000), if no arguments are provided.
- Configure 'gp-register-num' to the register the SP expects to receive the address of the boot info blob.
- Define 'boot-info' node and list the 'ffa_manifest'.
Limitations

- SPMC to SP boot information passing only.
- Parallel to firmware Handoff protocol.
- HOB and implementation defined boot information types not yet in use.
- SP Pkg impacts configuration of 'entrypoint-offset' in SP manifest and could be automated.
Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
شكرًا
danq
SP parsing manifest

```
<HF_ROOT>/test/hftest/hftest.py --out_partitions out/reference/secure_aem_v8a_fvp_vm_clang --log out/reference/kokoro_log --spmc out/reference/secure_aem_v8a_fvp_clang/hafnium.bin --driver=fvp --partitions_json test/vmapi/ffa_secure_partition_only/ffa_secure_partition_only_test.json --suite ffa_boot_info --test parse_fdt

/* Validate that SP can access its own FF-A manifest. */
TEST(ffa_boot_info, parse_fdt)
{
    struct ffa_boot_info_header* boot_info_header = get_boot_info_header();
    struct ffa_boot_info_desc* fdt_info;
    struct fdt fdt;
    struct fdt node root;
    void* fdt_ptr;
    size_t fdt_len;
    uint64_t ffa_version;

    fdt_info = get_boot_info_desc(boot_info_header, FFA_BOOT_INFO_TYPE_STD,
                                 FFA_BOOT_INFO_TYPE_ID_FDT);

    ASSERT_TRUE(fdt_info != NULL);

    HFTEST LOG("FF-A Manifest Address: %s", fdt_info->content);
    // MULTILINE(performance-no-int-to-ptr)
    fdt_ptr = (void*)fdt_info->content;

    ASSERT_TRUE(fdt_size_from_header(fdt_ptr, &fdt_len));
    ASSERT_TRUE(fdt Init from ptr(fdt, fdt ptr, fdt_len));
    EXPECT_TRUE(fdt_find_node(fdt, "/", &root));

    EXPECT_TRUE(fdt_is_compatible(&root, "arm.ffaboot-manifest-1.0"));
    EXPECT_TRUE(fdt_read_number(root, "ffaboot", &ffa_version));
    HFTEST LOG("FF-A Version: %x", ffa_version);

    ASSERT_EQ(ffa_version, MAKE_FFA_VERSION(1, 1));
```

INFO: Loading VM id 0x8001: ffa_secure_partition.
VERB Os: VM has 0 physical interrupts defined in manifest.
VERB Os: stream count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x6280000.
INFO: Hafnium initialisation completed
VERB Os: plat psci cpu resume: cpu mpidr 0x0 ON

[hello_world readline: get command line]

VM 8001: [hftest] FF-A Manifest Address: 0281000
VM 8001: [hftest] FF-A Version: 10001
VM 8001: [hftest] FINISHED

[hello_world readline: finished]
Build

• TF-A

```bash
make CROSS_COMPILE=aarch64-none-elf- \
SPD=spmd \
CTX_INCLUDE_EL2_REGS=1 \
ARM_ARCH_MINOR=5 \
BRANCH_PROTECTION=1 \
CTX_INCLUDE_PAUTH_REGS=1 \
PLAT=fvp DEBUG=1 \
BL33=../tf-a-tests/build/fvp/debug/tff.bin \
BL32=/home/joaalv01/Workspace/hafnium/out/reference/secure_aem_v8a_fvp_clang/hafnium.bin \
SP_LAYOUT_FILE=../tf-a-tests/build/fvp/debug/sp_layout.json \
all fip
```

• TF-A-Tests

```bash
make CROSS_COMPILE=aarch64-none-elf- \
PLAT=fvp \
DEBUG=1 \
TESTS=spm
```