TF-M Open Tech Forum

TF-M Performance Improvement in v1.5.0

Ken Liu & David Wang
9 Dec 2021
Introduction

• The design proposal was promoted in 2021 Spring.
  • Then went with times of prototyping and validation.
  • Got merged and then included in the 1.5.0 release.

• The main idea is to reduce the SPM execution time in the privileged mode.
  • To allow the interrupt preemption as quick as we can.

• It also involves new items in the project.
  • Synchronization.
  • Updated concept of SPM and NS Agent.
The classic implementation

- SVC
- SP1 thread
- PendSV
- Not interruptible
The current implementation (Isolation level 1) no IRQ
The current implementation (Isolation level 1) with IRQ
Summary

• Isolation level 2/3 to be fine-tuned
  • Now it still work under SVC-based implementation.

• Critical-section introduced into the design.
  • Those settings can be updated in the ISR.

• SPM function has the highest software priority
  • To avoid scheduling caused SPM API frame stacking.

• SPM needs a standalone working stack.
  • Re-use caller thread’s stack increased caller stack allocation size unexpectedly.
  • Can re-use TZ Trustzone Agent’s stack - Trustzone NS Agent is the NS interface of SPM.
Introduction

• Partition can have two runtime models
  • IPC model, which is similar to a process.
  • SFN model, which is similar to a library.

• The SFN model implementation
  • It is a model that contains SFN partitions and the NS Agent.
The SFN Model execution timeline
The SFN Model diagram
Summary

• Where the working stack is.
  • Under isolation level 1, NS Agent allocates the stack, and callees are working on it.
  • Several options for high-level isolation levels.

• Expand the IPC model
  • To make it run SFN partitions.
  • This avoids involving more ‘models’ into implementation.
Profiler Overview

- Initially developed as a tool for measuring PSA FF API cost and NS interrupt latency in TF-M
- Target is to make it generic and can be used for profiling TF-M.
- Defines a set of API/Macros to log the timing (timer tick or processor clock cycle) in lightweight to minimize the overhead from the profiler
- Supports different underlying HW – e.g. systick, Data Watchpoint and Trace (DWT), etc.
- Supports profiler overhead calibration
- Application/Host can dump the filtered data, analyze them, and print the report in desired format.
- Still working in progress for some minor issues and integration with TF-M/Test.
Performance Data for TF-M v1.4.0 and v1.5.0

• Initial TF-M performance data for watching
  • PSA FF API cost
    This is the cost of psa_connect/call/close. It’s measured with a dummy service.
  • NS interrupt latency
    It’s the non-interruptable time in TF-M from non-secure point of view. E.g. handler mode execution in SPE, critical section. It usually affects the real time performance of NS RTOS.

• Test platform: Musca S1
• Counter: DWT processor cycle counter
• Build configuration: IPC/SFN, isolation level 1, debug mode
• Toolchain: GNU Arm Embedded Toolchain 10.3-2021.07
**PSA FF API Cost**

*Note: As the Profiler and benchmarking test cases are still evolving, the numbers are subject to change.*
Non-Secure Interrupt Latency

*Note: As the Profiler and benchmarking test cases are still evolving, the numbers are subject to change.