EL3 GPT Fusing Discussion on various issues and approaches

TF-A Tech Forum

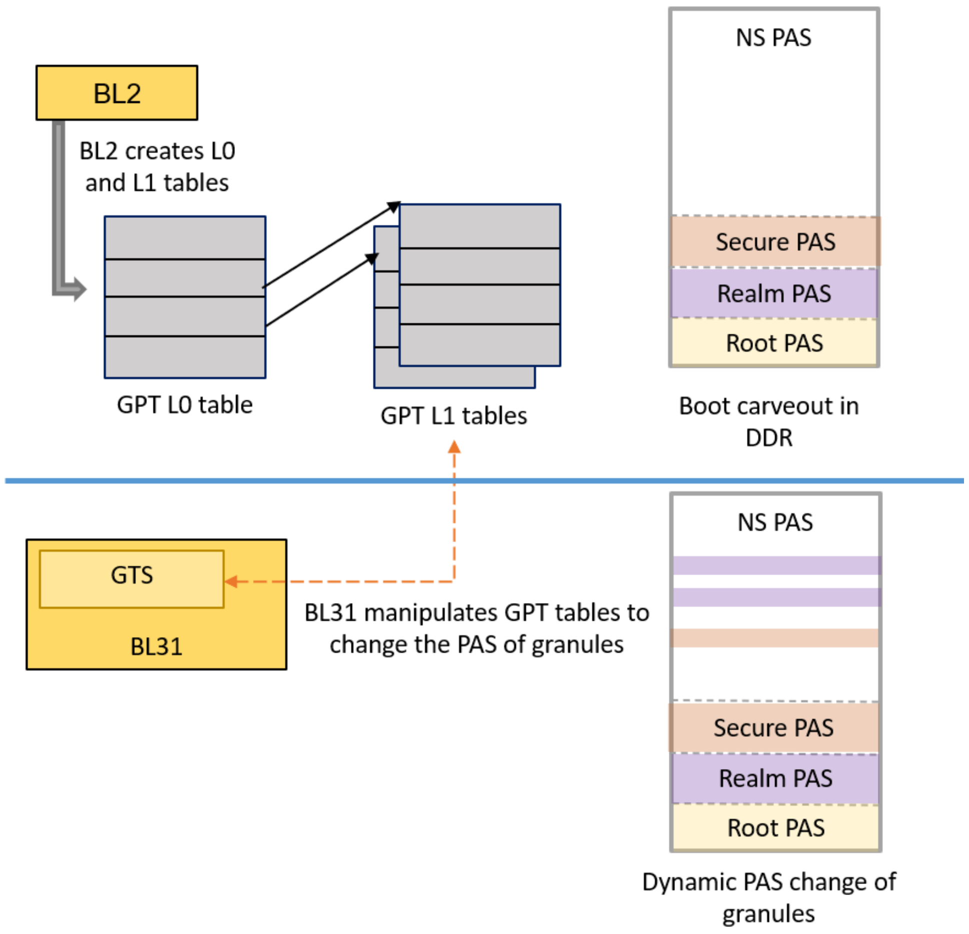
Soby Mathew & Alexei Fedorov
07-03-2024

Agenda

- Introduction
- Performance issue : TLB Shattering
  - PoC solutions explored – tradeoffs, inefficiency.
  - Inefficiency of 4K delegate - Need for block delegation.
  - Proposed upstream approach.
- Fine-grained locks for GPT access.
- Conclusion

Introduction to GPT

- The FEAT_RME introduces 2 new address spaces (total of 4)
  - The CPU accesses targets a physical address (PA) in one of the four physical address spaces.
  - The Granule Protection Check (GPC) is the mechanism by which accesses to those PA spaces are checked.
  - The in-memory structure that describes the association of physical granules with PA spaces is the Granule Protection Table (GPT)
    - There are 2 levels of lookup : L0 and L1 GPTs.
    - Level 1 GPT can be GPT Contiguous or GPT Granules descriptors.
    - The L0 table is in the SRAM (Shielded memory). The L1 table can be in DDR in Root PAS carveout.
  - The current TF-A only implements GPT granules descriptor. Every page has a GPI in the L1 table.



Performance issue with 4K GPI in L1

- EL3 does not use `Contiguous` field in GPT L1 today
- Certain workloads will expect 100% hit rate from the SMMU TBU. This is achieved by increasing the S1/S2 page size so that the application footprint can fit entirely within the TBU.
- Since EL3 currently does not fuse GPT entries (4K GPC granules only), it results in TBU misses under some workload footprints.
- **Requirement**: To give the application consistent HW performance, a given GPT entry must cover the same address range as the corresponding S2 translation (or S1 translation if operating in S1 only mode).
- This issue affects existing NS workloads utilizing SMMU.
- The working assumption is that TF-A (EL3 firmware) should be able to fuse to contiguous blocks opportunistically to avoid this problem.

Contig field encoding

Value	Meaning
0b00	Reserved
0b01	2MB
0b10	32MB
0b11	512MB

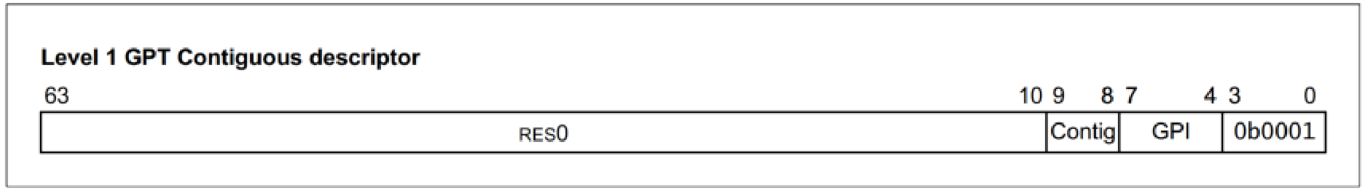


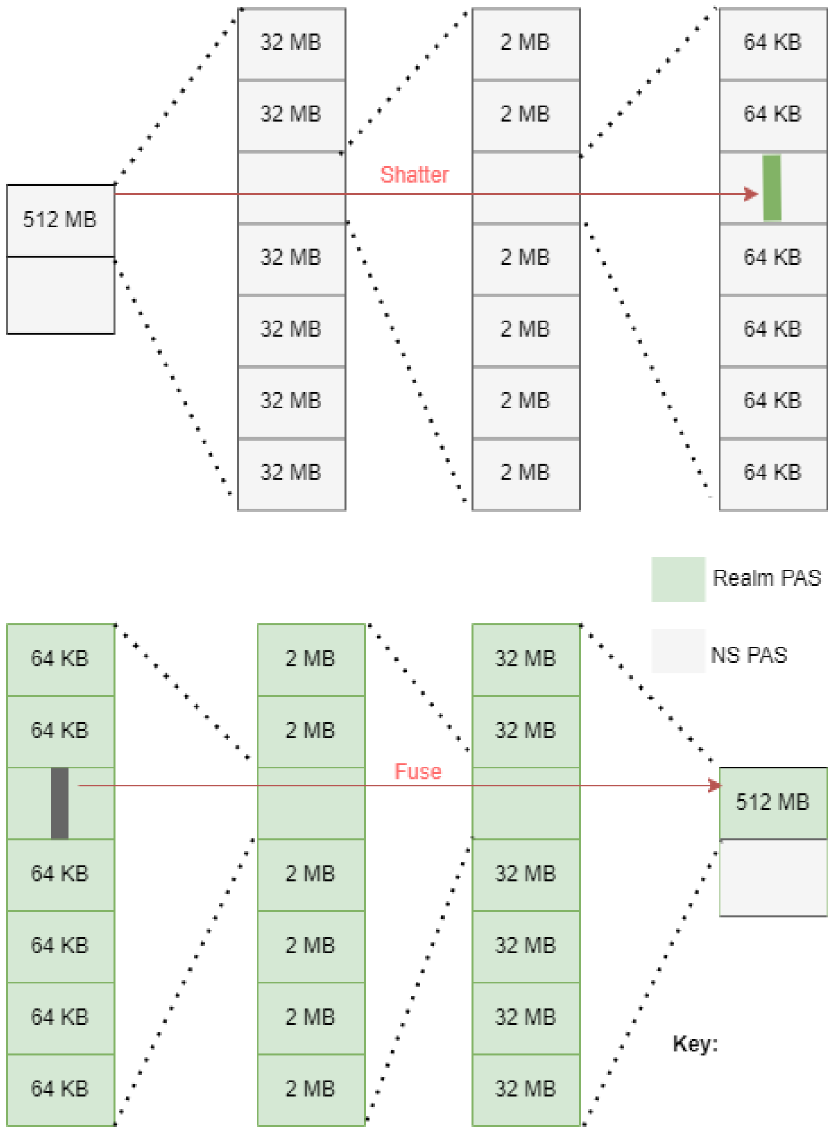
Figure D9-4 Level 1 GPT Contiguous descriptor format

EL3 Prototyping 1 (Brute force)

- Fuse and Shatter “transparently” in EL3.
  - Use of `Contig` field in GPT level 1.
  - Creation of Level 0 block is not considered.
- Every delegate (NS -> Realm) potentially involves a Fuse or Shatter.
  - Similarly, on undelegate (Realm -> NS).
- The figure shows 2 cases :
  - Brute force without any tracking meta data is inefficient for fuse.
  - Need to hold lock at largest fuse block level exacerbate the wait times.

Case 1: Delegate the first Realm GPI

Case 2: Delegate the last outstanding NS GPI

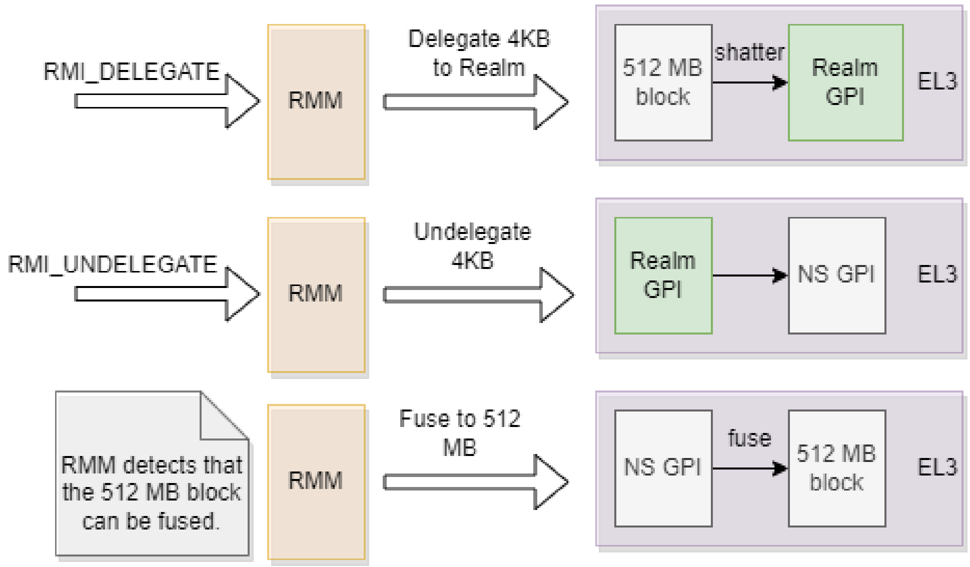


EL3 prototype 2 (with Metadata in EL3)

- Maintaining some meta data in EL3 and keeping track avoids the lookup cost of Brute force.
  - We avoid unsuccessful lookups.
  - Intermediate block creation can be skipped if a higher contig block is possible.
- Memory cost of ~1KB for every 1 GB of data to track delegation per world.
  - ~2KB if both Secure & Realm world delegation.
- This is too expensive to keep in SRAM
  - CCA Security model mandates any EL3 critical data should be in Shielded Memory (SRAM).
  - Deal breaker for this scheme AFAICS.

EL3 prototype 3 (with Metadata in RMM)

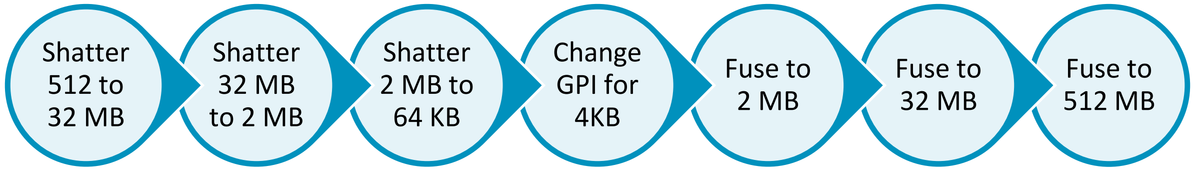
- EL3 provides GPT fuse mechanism but does not make the policy decision to fuse.
  - EL3 to create GPT L1 in fused state (initial boot state).
  - EL3 to shatter L1 tables as required by delegation (But no fuse).
  - EL3 to receive fuse hint from client.
    - On receiving hint, EL3 validates the fuse block and proceeds to fuse the hinted size.
- Client (RMM in the case of Realm PAS) to keep tracking metadata information and provides fuse hint on detecting that a fuse can be done.
  - Fuse policy determined by client based on requirement including aggressiveness of fuse.
- A similar scheme may be possible for Secure world.
  - EL3 can provide brute force method if hint is not possible for some reason.



The sequence when delegating a 4KB and un-delegating a single 4KB page.

The inefficiency of Fusing with 4K delegation

- Assume a 512 MB block needs to be delegated to realm world.
  - Regardless of the approach, this inefficiency is present for 4K based delegation.



- The best-case scenario is the 512 MB NS block switches to 512 MB Realm in one go.
  - The delegate and undelegate requests are naturally aligned to block sizes.
  - If most or all the requests are “block” request, we avoid the issues of brute force as well.
- RMM ABI for delegate are 4K based.
  - Needs change in RMM ABI.
  - Without change in ABI , we would have this inefficiency which may affect the Realm launch and teardown times.

Proposed upstream approach

- The dilemma:
  - What is the benefit of fusing beyond 2 MB block?
  - Is Prototype 3 measurably better than Prototype 1?
  - Without block mapping ABI, what is the measurable impact in Realm launch times?
  - If RMM ABI does introduce Block mapping, then issues mentioned for Prototype1 are insignificant?
- The upstream approach:
  - Introduce the approach in prototype 1 but limit the block size to 2MB by default.
    - TF-A provides configurable option to increase block size up to 512 MB.
  - Once we get feedback on some of the questions, then we can pivot in upstream
    - Need a solution that works for different workloads (both NS and Realm).

Fine Grained Locks for GPT access

- EL3 needs to access GPT in a mutually exclusive manner.
  - The locking scheme need to be fine grained to allow multiple threads to modify GPT.
- The smallest lock granularity need to be at the highest fuse block size level without hand-over-hand locking
  - This would mean a lock at 512 MB granularity or higher.
  - A Hand-over-hand locking scheme while shattering to smaller sizes is possible, but the performance improvement is not clear and the memory costs are too high.
    - Hand-over-hand locking scheme not possible for brute force implementation.
- We have implemented a bit lock per 512 MB => 256 bytes of SRAM for 1 TB of PA space.
  - Device Assignment support for RME implies the locking scheme has to cover the PA space rather than DDR size.
  - The implementation has the following behavior :

    When locking: acquire a byte lock to set a bit.

    When unlocking : acquire a byte lock to clear a bit.

Conclusion

- The upstream patch is available here for review : https://review.trustedfirmware.org/c/TF-A/trusted-firmware-a/+/26674
- Any feedback on performance or input on the dilemmas posed will be helpful.

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు