openEuler

SMMU驱动性能优化

雷镇

目录

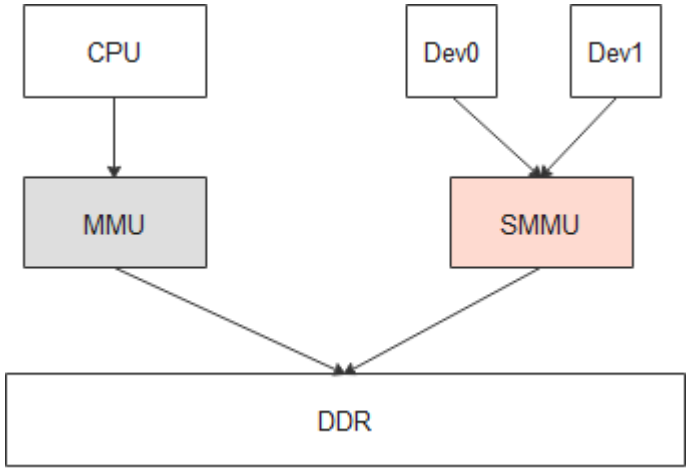
CONTENT

openEuler

SMMU简介

SMMU的作用

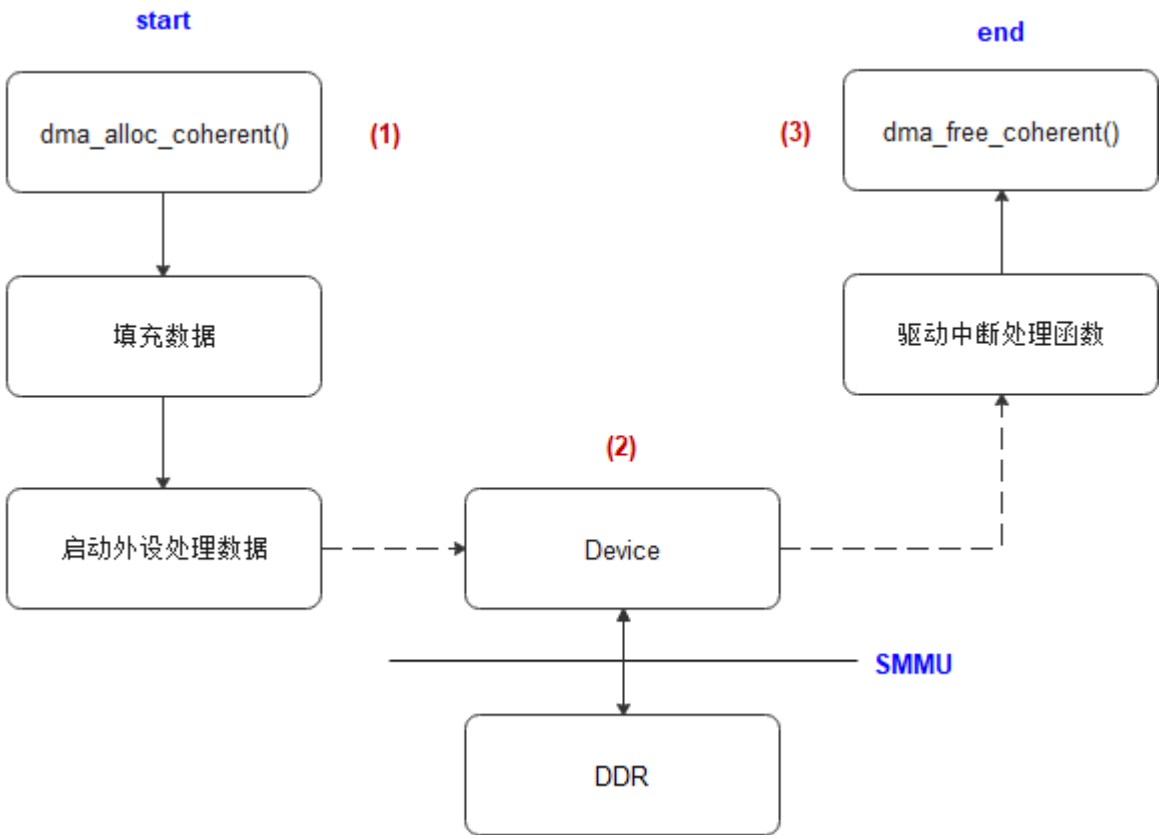
- IO地址转换
  - 资源隔离，外设只能访问驱动成功分配的内存。
  - Scatter/gather，将离散的物理内存映射成连续的IOVA区间。
  - SVM，实现外设和用户态进程共享同一张页表。
  - 支持32位外设访问64位地址空间。
- 设备隔离
  - 没有匹配驱动的外设无法访问系统内存。



openEuler

SMMU简介

内核驱动运作基本流程



openEuler

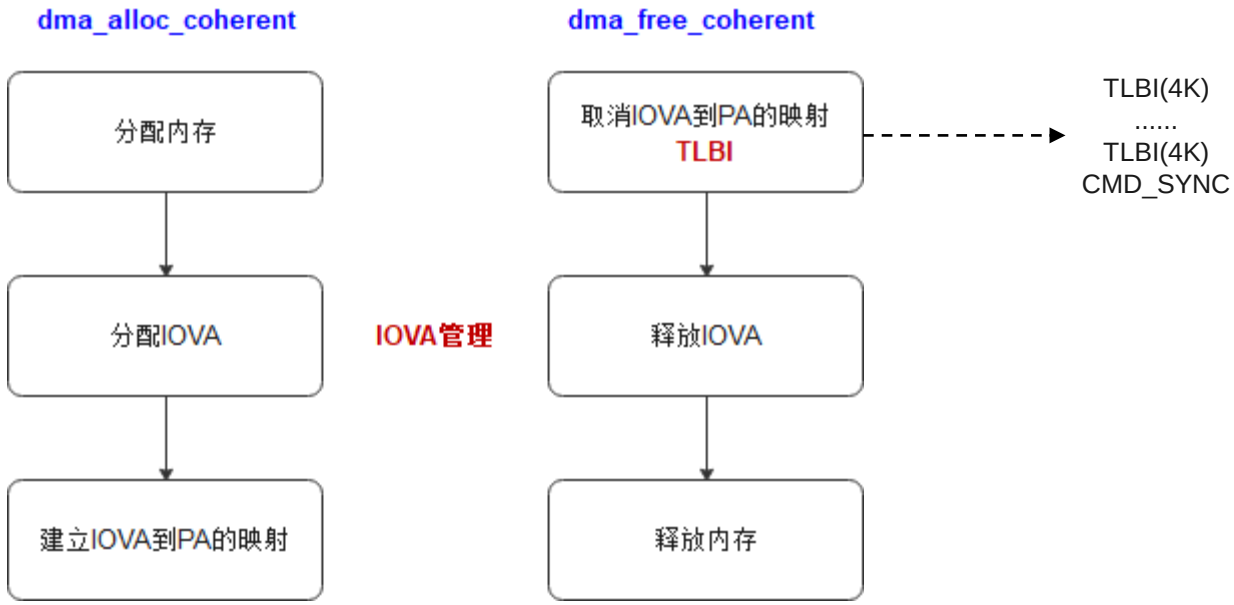
SMMU简介

DMA内存分配和释放函数展开

性能瓶颈主要集中在：

1、TLBI+SYNC命令的插入和执行

2、IOVA的申请和释放

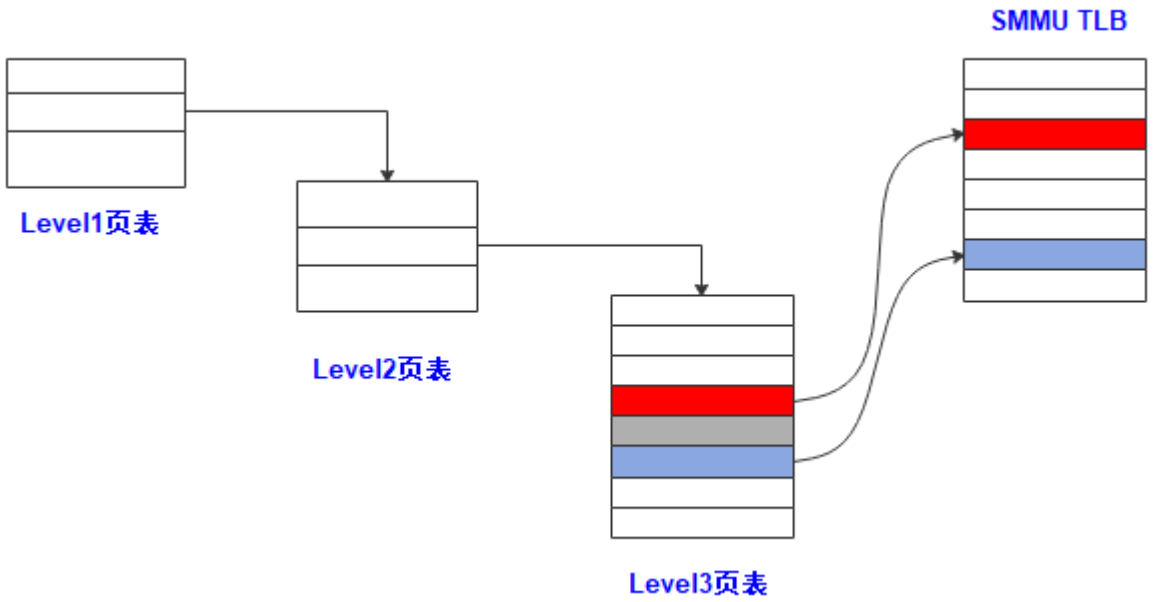


openEuler

SMMU简介

TLB的作用

TLB用于缓存页表中pte项，提升地址转换性能。



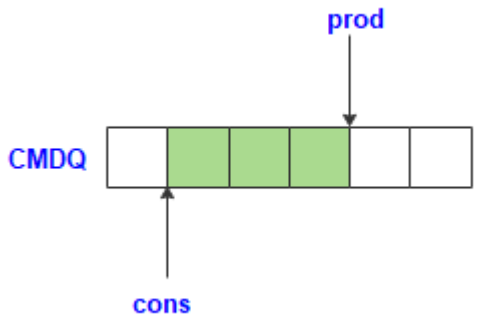
openEuler

SMMU简介

CMDQ的运作机制

每个SMMU都拥有一个循环队列CMDQ，其大小由软件配置，但硬件限制元素数必须为2的指数幂且有上限（最大219）。Prod为生产者指针，由软件控制，每当填入一条命令后，prod++；Cons为消费者指针，由硬件控制，每当取走一条命令时，cons++。当prod或cons出现绕卷时，切换相应的wrap位。当prod和cons相等时，队列为空；当prod和cons索引（index）相等，但wrap不相等时，队列满。

Cons的更新只是说明硬件取走了命令，但并意味着硬件执行完了命令。为了确保命令执行完成，应插入一条SYNC命令。SYNC确保前面所有的命令都执行完了，才更新cons。



openEuler

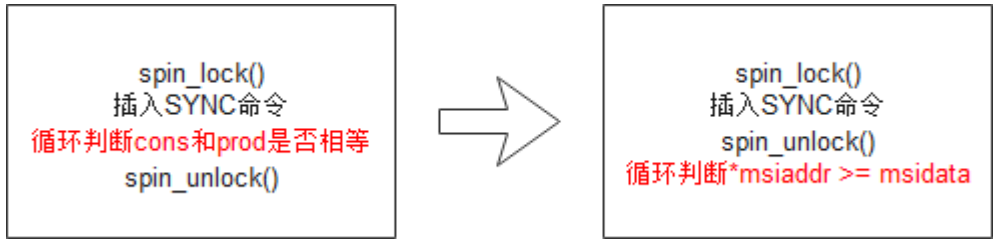
SMMU软件优化

SYNC命令优化(1)

37de98f8f1cf330 iommu/arm-smmu-v3: Use CMD_SYNC completion MSI

将等待SYNC命令执行完成移到锁保护外。

若SYNC配置为MSI模式，则每当SYNC命令执行完成时，硬件会将msidata值写入msiaddr处。补丁令所有的SYNC命令共享同一个msiaddr，而msidata值每次单调递增。因为命令是顺序执行的，因此只会出现大号的msidata覆盖小号的msidata的情况。各核判断从msiaddr处读取的值是否大于等于自身SYNC命令对应的msidata值，即可得知是否执行完成。



SMMU软件优化

SYNC命令优化(1)-优化效果

https://www.mail-archive.com/iommu@lists.linux-foundation.org/msg19756.html

I tested this on QDF2400 hardware which supports MSI as a CMD_SYNC
completion signal. As with Thunder's "performance optimization" series,
I evaluated the patches using FIO with 4 NVME drives connected to a
single SMMU. Here's how they compared:

FIO - 512k blocksize / io-depth 32 / 1 thread per drive
 Baseline 4.13-rc1 w/SMMU enabled: 25% of SMMU bypass performance
 Baseline + Thunder Patch 1   : 28%
 Baseline + CMD_SYNC Optimization: 36%
 Baseline + Thunder Patches 2-5 : 86%
 Baseline + Thunder Patches 1-5 : 100% [!!]

Seems like it would probably be worthwhile to implement this for the
non-MSI case also. Let me know if there are other workloads you're
particularly interested in, and I'll try to get those tested too.

openEuler

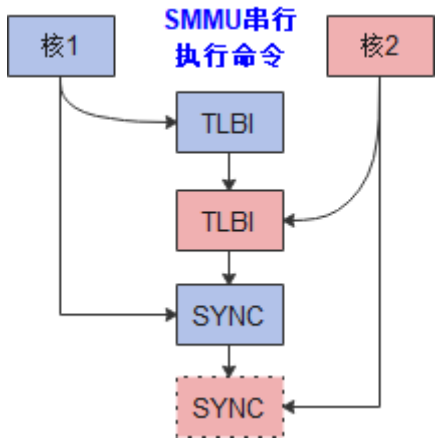
SMMU软件优化

SYNC命令优化(2)

901510ee32f7 iommu/arm-smmu-v3: Avoid back-to-back CMD_SYNC operations

消除背靠背的SYNC命令。

由于各核插入命令的并发性，可能会出现两个SYNC命令背靠背的情况。显然此时执行第二个SYNC命令是多余的，应省掉。



openEuler

SMMU软件优化

SYNC命令优化(2)-优化效果

在压力场景下可以减少1/3的SYNC命令数。

```
has been shown to improve IO performance under pressure, where the
number of SYNC operations reduces by about a third:

        CMD_SYNCs reduced:      19542181
        CMD_SYNCs total:        58098548        (include reduced)
        CMDs total:             116197099       (TLBI:SYNC about 1:1)
```

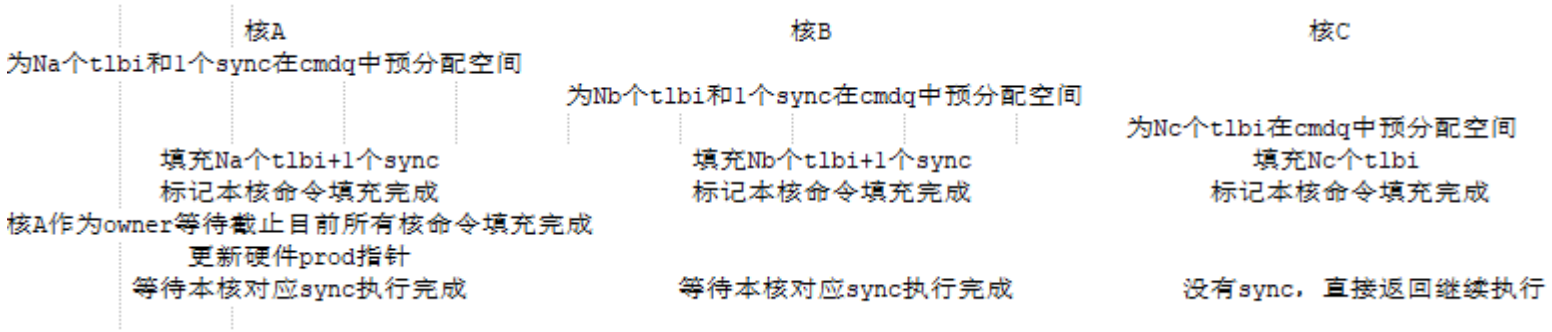
openEuler

SMMU软件优化

CMD并发插入

587e6c10a7ce iommu/arm-smmu-v3: Reduce contention during command-queue insertion

通过预分配cmdq空间，使得各核可以并发地填充命令，从而减小了多核间共享资源冲突的概率，提升了并发度。



openEuler

SMMU软件优化

CMD并发插入-优化效果

100G网卡对接场景性能提升89.2%，达到满规格；NVME盘MySQL场景性能提升29.4%，达到满规格的93.4%。

100G网卡对接
passthrough
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec

non-strict
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec

v5.3
[SUM] 0.0-10.0 sec 56.0 GBytes 48.1 Gbits/sec
[SUM] 0.0-10.0 sec 56.5 GBytes 48.5 Gbits/sec
[SUM] 0.0-10.0 sec 61.3 GBytes 52.7 Gbits/sec

v5.4 合入CMD并发插入
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec
[SUM] 0.0-10.0 sec 110 GBytes 94.2 Gbits/sec

NVME盘MySQL数据库
passthrough(before) avg=665216.65
666971.01
667417.98
661260.96

non-strict(before) avg=656581.53
655804.85
657358.2

strict(before) avg=480137.34
472700.08
487574.6

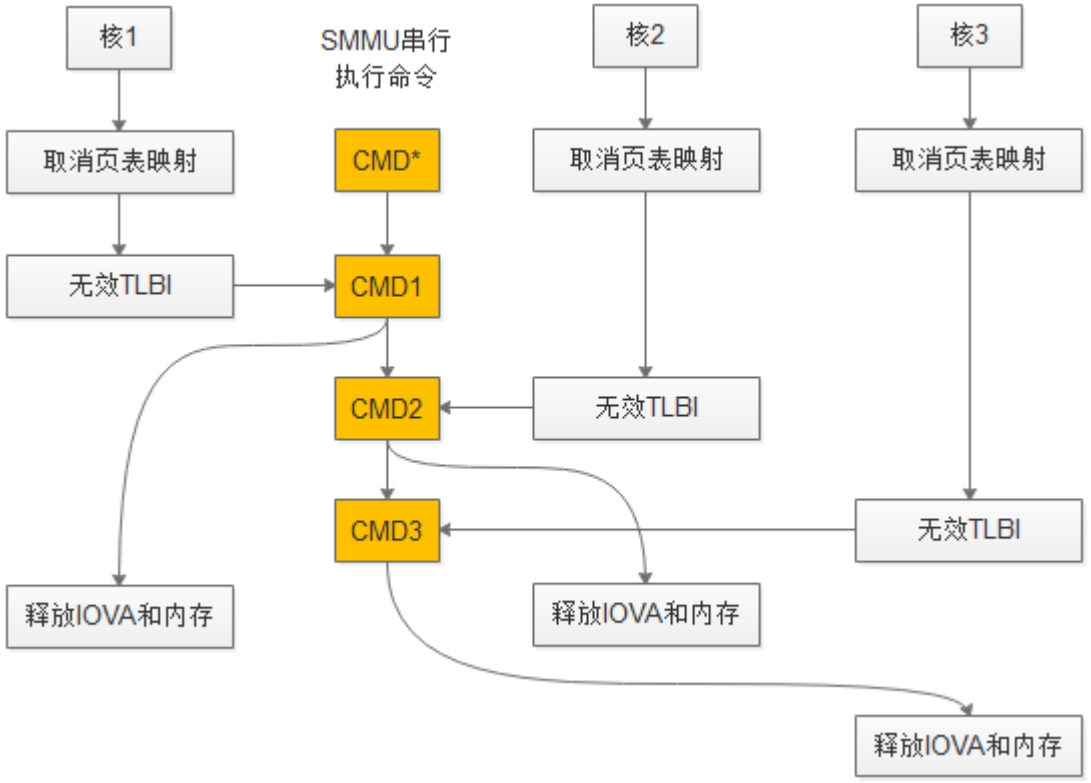
strict(after) avg=621123.69
611455.33
628005.77
623909.96

openEuler

SMMU软件优化

strict模式的局限性

假定各核可以快速并发完成各个动作，但IOMMU硬件需要串行完成CMD1、CMD2、CMD3。另假定SMMU硬件完成CMDx的时间为x，那么一秒内可以完成的最大命令数为1/x。如果网卡每收发4K的数据需要各执行一次unmap，那么最大的数据性能值为4K*(1/x)。



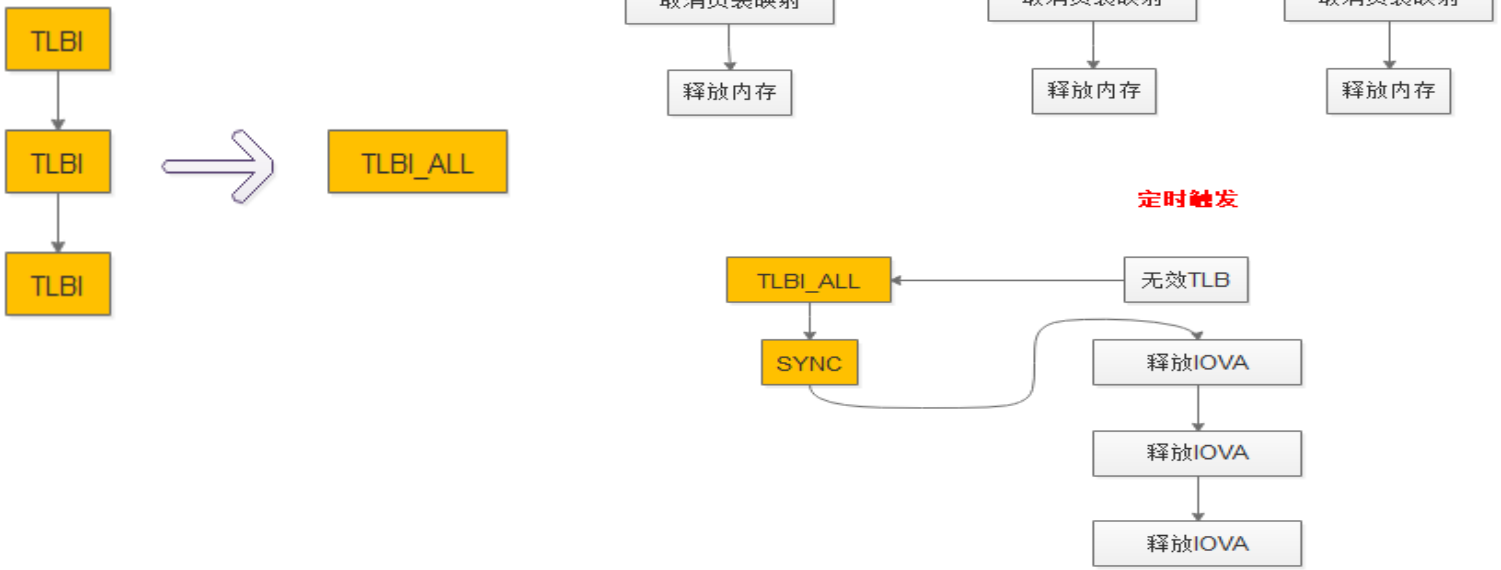
openEuler

SMMU软件优化

non-strict模式

9662b99a19ab iommu/arm-smmu-v3: Add support for non-strict mode

non-strict模式延迟TLB的无效和IOVA的释放，因此各核的unmap可以快速完成。同时将单位时间内所有的N个TLBI操作，替换成一个TLBI_ALL命令。



openEuler

SMMU软件优化

non-strict模式的缺陷

1）安全性

由于内存已提前释放，这些释放掉的内存可以被其它应用申请使用。而TLB中缓存的页表映射并没有立即被无效，在TLBI_ALL执行之前，非法驱动或外设可基于原先的IOVA访问原先映射的物理内存，如果这些物理内存已被其它应用拿去使用，那么非法者在机会窗中可以看到这些数据。

2）性能瞬间陡降

TLBI_ALL会无效所有的TLB缓存，包括非unmap掉的IOVA映射，导致全部需要重新预取。此外，IOVA集中释放会导致操作核的CPU占用率短暂维持高位。

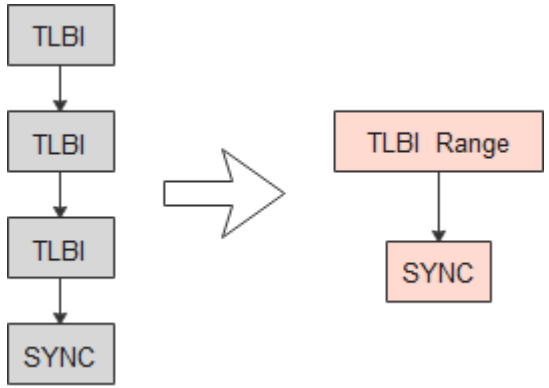
SMMU硬件优化

TLBI Range

6a481a95d4c1 iommu/arm-smmu-v3: Add SMMUv3.2 range invalidation support

SMMUv3.2开始支持。在原TLBI命令的基础上扩展了NUM[16:12] 和SCALE[24:20]两个字段，单个TLBI Range命令可以无效一片IOVA区间。

Range = ((NUM+1)*2^{SCALE})*Translation_Granule_Size

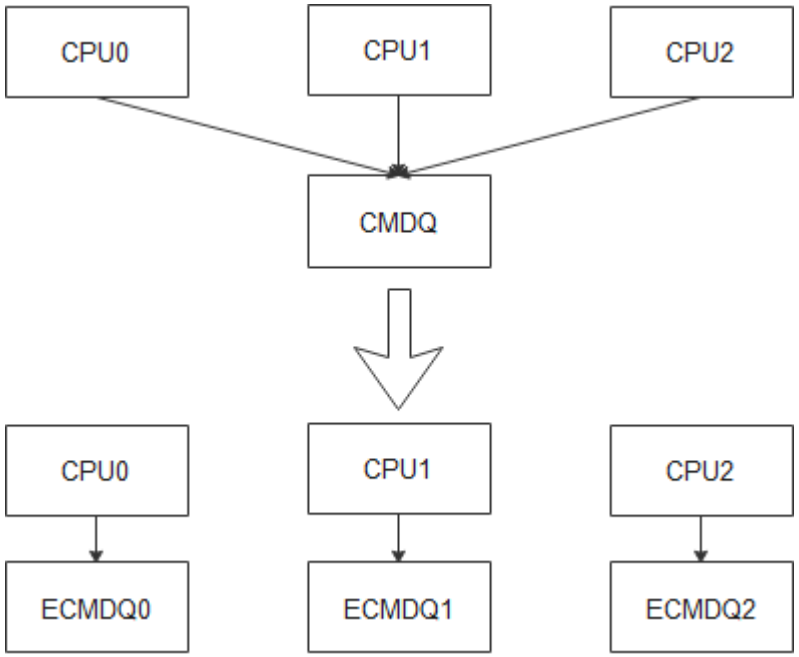


openEuler

SMMU硬件优化

Ehanced CMDQ

SMMUv3.3开始支持。硬件提供一组ECMDQ，最理想的情况每个核可以独占一个ECMDQ。各核使用自己专用的ECMDQ插入命令，无需互斥。硬件甚至可能实现各个ECMDQ的命令并行执行。



openEuler

IOVA优化

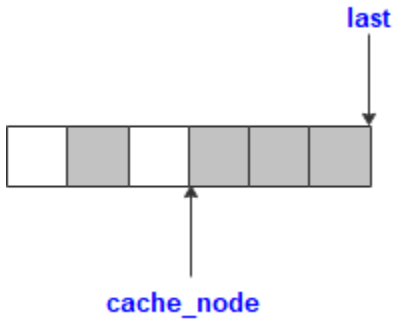
64位IOVA优化

https://lkml.org/lkml/2017/9/19/489

https://lists.linuxfoundation.org/pipermail/iommu/2017-March/021096.html

软件只维护了dma32的cache_node，dma64分配时总是基于last指针开始搜索。当已经分配了大量dma64的IOVA而没有释放时，就会导致从last开始需要遍历完成千上万的已分配iova结点，才能找到空闲的IOVA空间。

解决思路就是参照dma32的cache_node，为dma32和dma64各维护一个cache_node指针。每当分配时，优先基于cache_node指向的位置搜索空闲IOVA；每当释放IOVA时，适当调整cache_node值。



openEuler

IOVA优化

64位IOVA优化效果

https://lists.linuxfoundation.org/pipermail/iommu/2017-March/021095.html

```
Below it's the performance data before and after my patch series:
(before)$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 35898
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.2 sec  7.88 MBytes  6.48 Mbits/sec
[  5] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 35900
[  5]  0.0-10.3 sec  7.88 MBytes  6.43 Mbits/sec
[  4] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 35902
[  4]  0.0-10.3 sec  7.88 MBytes  6.43 Mbits/sec

(after)$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 36330
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  1.09 GBytes   933 Mbits/sec
[  5] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 36332
[  5]  0.0-10.0 sec  1.10 GBytes   939 Mbits/sec
[  4] local 192.168.1.106 port 5001 connected with 192.168.1.198 port 36334
[  4]  0.0-10.0 sec  1.10 GBytes   938 Mbits/sec
```

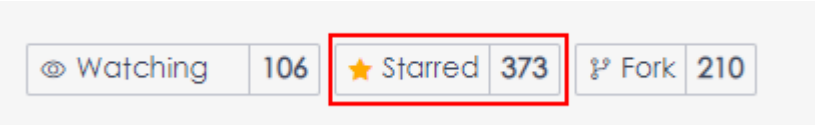
openEuler

✓ openEuler kernel gitee 仓库

源代码仓库

https://gitee.com/openeuler/kernel

欢迎大家多多 Star，多多参与社区开发，多多贡献补丁。



✓ maillist、issue、bugzilla

可以通过邮件列表、issue、bugzilla 参与社区讨论

欢迎大家多多讨论问题，发现问题多提 issue、bugzilla

https://gitee.com/openeuler/kernel/issues

https://bugzilla.openeuler.org

kernel@openeuler.org

✓ openEuler kernel SIG 微信技术交流群

请扫描右方二维码添加小助手微信

或者直接添加小助手微信（微信号：openeuler-kernel）

备注“交流群”或“技术交流”

加入 openEuler kernel SIG 技术交流群

技术交流

openEuler

Thank you

openEuler