bmcv\_image\_vpp\_convert 性能分析

目前，bmcv库封装了vpp的调用，用户不用直接操作vpp寄存器就可以使用vpp进行后处理。主要bmcv api为bmcv\_image\_vpp\_convert。本文将展示典型场景下该函数的性能。

芯片bm1684中的vpp core，时间可以近似认为

其中 与DDR带宽，src数据量有关，与DDR带宽和dst数据量有关。而则相对固定，与输入数据量，输入格式有关（如果输入格式为COMPRESSED的话，为了解压缩，需要耗费更多的vpp计算时间）。以4张1920x1080的nv12转rgb图片为例。

|  |  |  |
| --- | --- | --- |
| 源图片位置 | 目的图片位置 | 时间 |
| DDR1 | DDR1 | 3.67ms |
| DDR1 | DDR0 | 3.68ms |

以上测试在fast模式下进行。从结果上看，DDR0 / DDR1位置对于最终结果影响甚微，如果没有产生ddr的阻塞，可以认为dst位置对于最终结果没有太大影响。

计算src的数据量，src为NV12格式，总共的大小为1920x1080x4x1.5 = 12441600 Byte，fast模式下ddr的带宽为16.664G Byte / s, 可以计算出理想情况下。

同理，计算dst的数据量，dst为RGB\_PLANAR格式，大小为1920x1080x3x4 = 24883200 Byte，理想情况下

假设以上条件都是理想的，vpp运算时间为3.67ms – 1.4932ms – 0.7466ms = 1.4302ms。

现在考虑另一种情况，假设输入src为4张1920x1080的NV12格式图片，输出resize为4张960x540的RGB图片。理想情况下，不变，仍为，由于数据量变为原先的1/4, 因此为1.4932ms / 4 = 0.3733ms。而不变，计算出理论耗时为0.7466 + 1.4302 + 0.3733 = 2.5501ms。实际测试结果为2.554ms，与理论计算结果相符。

下表为soc fast mode下单线程调用vpp\_convert函数各个src数据量情况下的实测性能表现。



以上转换src 格式为FORMAT\_NV12, dst格式为FORMAT\_BGR\_PACKED，src memory在ddr1上，且src与dst stride均64对齐减少了多余的width align，提升运行速度。