Rockchip RV1126 UVC延时优化

文件标识：RK-KF-YF-540

发布版本：V1.0.0

日期：2021-08-04

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档旨在指导工程师如何使用Rockchip RV1126 Linux 平台UVC延时优化功能方案。

产品版本

芯片名称	内核版本
RV1126/1109	Linux 4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	HuangJC、CW、LXH、TZB、LQH	2021-08-04	初始版本

目录

Rockchip RV1126 UVC延时优化

- 简介
- UVC延时分解
  - 分解框图
  - 延时打印方法
- UVC延时优化
  - sensor曝光时间优化
  - cif/isp/ispp耗时优化
    - isp直通优化方案
    - line配置并行优化方案
      - cif优化
      - isp/ispp优化
  - rga处理和调度耗时优化
  - mpp编码耗时优化
  - usb处理和调度耗时优化

简介

Rockchip RV1126 Linux平台支持UVC全功能预览，对智慧屏摄像头、直播摄像头、会议摄像头等通用UVC类产品，延时指标是重点关注项目，本文档可以指导客户如何根据自身产品配置优化延时指标，协助改善产品用户体验和提高竞争力。

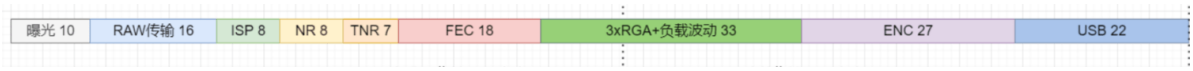
UVC延时分解

分解框图

UVC预览延时主要分device端处理：sensor出流 、isp处理、ispp处理、编码、usb处理送流，和host端处理：usb收流、应用取流解码处理、显示处理。

RV1126 UVC产品主要作为device端，所以Host端处理不在本文档优化范围。

device端延时分解如下（以SC500AI为例）：



其中：

1. sensor曝光时间：不同sensor不同光线环境不同，一般是10-33ms，可通过IQ配置控制范围，需要和画质要求权衡；

2. raw传输：一般sensor配置是接cif进入，vicap取流，这块为mipi传输耗时，同样和sensor本身有关，一般16-33ms；
3. isp处理：isp处理耗时，不同sensor和IQ配置不同；
4. ispp处理：ispp处理耗时，主要是nr、tnr、fec模块耗时，一般模组不需要开fec；
5. rga处理和调度：应用层可能定制的rga预处理图像数据操作导致的耗时，如裁剪、缩放等；
6. uvc编码处理：uvc调用编码器根据Host端要求的格式编码处理耗时，不同格式编码耗时不同，同分辨率下mjpeg格式耗时较长；
7. usb处理和传输：usb驱动分帧编码和传输到Host端耗时。

以SC500AI 这颗sensor（1440p）接入的产品为例，预览1440p@30fps mjpeg格式图像耗时数据如下，可以看到是比较长的，和公版对比需要至少优化到100ms以内。

```
Device端总耗时：10 + 16 + 8 + 8 + 7 + 18 + 33 + 27 + 22 = 142ms
```

延时打印方法

本章介绍RV1126平台各模块延时统计方法，需要在预览状态下打印。

1. sensor曝光时间打印方法：

   机器RkLunch.sh脚本中添加一行，开启AIQ的AE打印：

   ```
   export persist_camera_engine_log=0x1ff3

   uvc预览时串口就会多出如下AE打印：
   [00:18:32.026984][AEC]:XCAM DEBUG rk_aiq_ae_algo.cpp:7200: >>> Framenum=156
   Cur Piris=512,
   Sgain=1.000000,Stime=0.002196,mgain=1.687500,mtime=0.029992,lgain=0.000000,l
   time=0.000000
   ```

   可以看到上面EVB公版(os04a sensor)配置下打印的2帧(HDR)曝光时间分别为：2.2ms和30ms(Stime=0.002196 ,mtime=0.029992),非HDR只有一个time打印有值表示曝光时间。

2. mipi传输（cif）耗时打印方法：

   可以通过cat /proc/uvcinfo打印csi transport delay耗时31ms（EVB公版），如下：

   ```
   [root@RV1126_RV1109:/]# cat proc/uvcinfo
   vc on: 1 1 0 0
   csi sequence: 2263 2263 0 0
   csi transport delay: 30224003 31716170 0 0  #mipi传输耗时，这里开hdr 有2帧
   stream sequence: 2264 2264 0 0
   stream sequence map: 2263 2263 0 0
   stream vb2 done: 25604586 25601086 0 0
   usb transport bytes: 44650
   usb transport delay: 2398084  #usb传输延时打印 2.3ms传完
   ```

   后面打上cif优化补丁后也有cif节点可以直接查看耗时。

3. isp/ispp耗时打印方法：

   ```
   [root@RV1126_RV1109:/]# cat proc/rkispp*
   rkispp0    Version:v01.06.00
   clk_ispp   491519999
   aclk_ispp  491519999
   hclk_ispp  245760000
   Interrupt  Cnt:127584 ErrCnt:0
   ```

```
Input      rkisp0 Format:FBC420 Size:2688x1520 (frame:42527 rate:33ms
delay:6ms)  #优化前delay：15ms
Output     rkispp_scale0 Format:NV12 Size:1920x1080 (frame:42527 rate:33ms
delay:18ms) #优化前delay：37ms
TNR        ON(0xf00000f) (mode: 3to1) (global gain: disable) (frame:42527
time:8ms idle) CNT:0x0 STATE:0x1e000000
NR         ON(0x57) (external gain: enable) (frame:42527 time:6ms idle)
0x5f0:0x0 0x5f4:0x0
SHARP      ON(0x19) (YNR input filter: ON) (local ratio: OFF) 0x630:0x0
FEC        OFF(0x2) (frame:0 time:0ms idle) 0xc90:0x0
ORB        OFF(0x0)
Monitor    ON Cnt:0
```

其中isp的耗时可以查看Input这一行的delay打印，isp+ispp具体一路的总耗时可以看Output这一行的delay打印.

4. 编码及uvc调度传输耗时打印方法：

```
[root@RV1126_RV1109:/]#touch /tmp/uvc_use_time && sleep 1 && rm
/tmp/uvc_use_time

mpp[667]: [shm_control_uvc][sendUVCBuffer]:isp->aiserver seq:1883 latency
time:20302 us, 20 ms
[uvc_ipc][recvUVCBuffer]:isp->aiserver->ipc seq: 1883 latency time:20968 us,
20 ms
[uvc_app][test_mpp_run]:isp->aiserver->ipc->mpp_get_buf seq:1883 latency
time:21380 us, 21 ms
[uvc_app][test_mpp_run]:mpp_enc+getbuf time:12 ms, try_count:15
[uvc_app][test_mpp_run]:isp->aiserver->ipc->mpp_enc_ok latency time:32054
us, 32 ms
[uvc_app][uvc_delay_time_calcu_after_get]:isp->mpp->usb_ready seq:1883
latency time:33061 us, 33 ms
[uvc_app][uvc_delay_time_calcu_before_get]:isp->mpp->usb_ready->usb_send_ok
seq:1883 latency time:35726 us, 35 ms
```

ispp上来的buf带时间戳信息，通过应用层的打印可以统计后续阶段的耗时，如上EVB（带优化）的打印耗时：

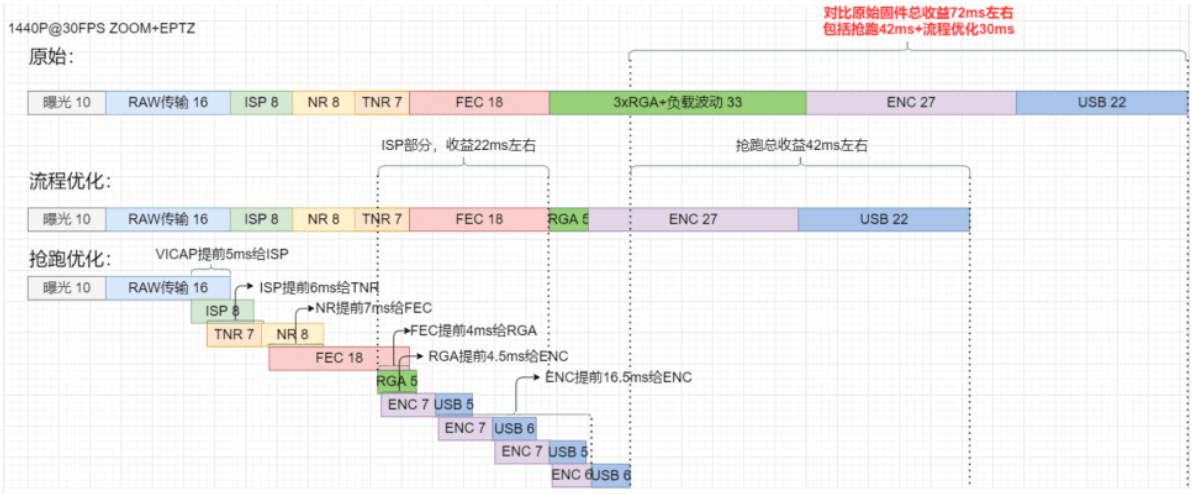
编码耗时：mpp_enc+getbuf time:12 ms

uvc传输耗时：35 - 32 =3 ms

UVC延时优化

以SC500AI为例，优化方案如下图，主要采用并行抢跑方案，优化后device端总延时：

```
10ms(曝光) + 11ms(传输)+ 57ms(vicap dqbuf->usb qbuf#4) + 7ms(usb#4) = 85ms左右
```



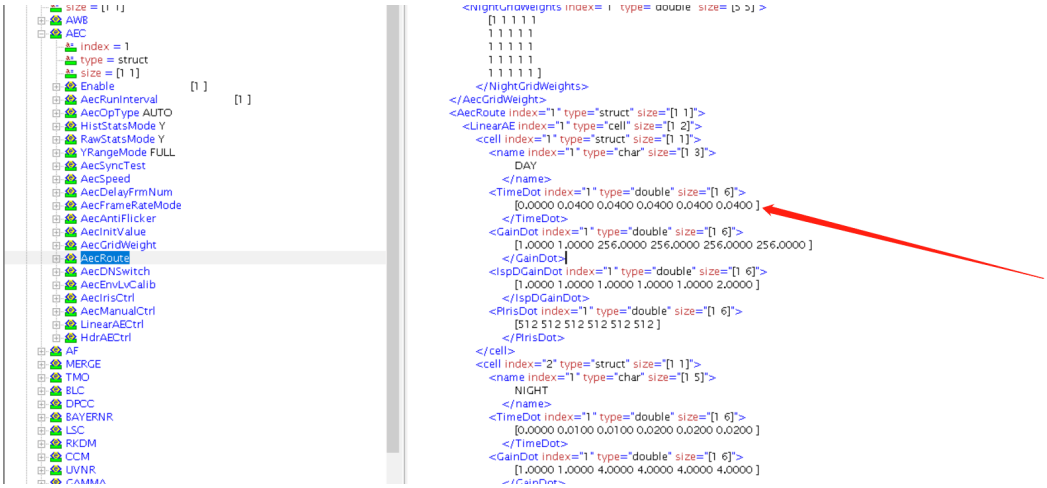
下面介绍针对device端各耗时模块的具体优化方案，供其它sensor和软件配置产品参考使用。

sensor曝光时间优化

上述例子中sensor曝光时间本身已经很短，没有优化空间，若其它sensor配置可以通过上面方法通过打印确认当前的曝光时间。

降低曝光时间方法：

1. 在iq文件中搜索LinearAE，下的TimeDot，将最大曝光时间缩短;
2. 非HDR一般改DAY项目中的TimeDot即可，开HDR的配置MTimeDot一般也要改，通过AE打印观察是否生效即可。



cif/isp/ispp耗时优化

该组合部分和产品dts中配置的sensor接入方式有关，主要分为走isp_in接入和cif接入：

isp直通优化方案

要走isp_in接入方式，可以使用走isp直通方式优化延时，具体方法为：

- 编译kernel:

  git revert 000c394b31f9a95de9ac17e4335b892876bd2e60 编译kernel，单独烧写kernel目录下生成的zboot.img。

  上述默认将ai camera dts对应产品改成isp in接入方式，客户若用自己的dts，参考修改为isp in接法配置即可。

- 直接修改设备文件:

  adb shell; vi /etc/aicamera.sh; 启动ispserver前添加export normal_no_read_back=1

验收方法：通过预览时cat /proc/rkispp*确认Input一行中delay打印为0，说明修改成功。

收益：看具体sensor产品情况，一般8ms以上收益。

局限性：直通方式无法启用DeHaze/Enhance模式，如果产品原先画质有开去雾的会受影响，客户需要自行权衡画质损失。

line配置并行优化方案

该部分优化方案主要是根据按行处理的特性，处理一帧数据时，前一级模块提前送数据，启动后一级模块的处理，达到并行效果，优化总耗时。所以该方案需要具体产品实测来优化line设置，一般会有15ms以上收益。

为方便耗时统计，可以按下面的方式修改时间戳，从cif开始计算，通过节点可直接看出3个阶段总耗时：

```
diff --git a/drivers/media/platform/rockchip/isp/csi.c
b/drivers/media/platform/rockchip/isp/csi.c
index 631c1de..56ef0bf 100644
--- a/drivers/media/platform/rockchip/isp/csi.c
+++ b/drivers/media/platform/rockchip/isp/csi.c
@@ -645,7 +645,7 @@ static void rkisp_dev_trigger_handle(struct rkisp_device
*dev, u32 cmd)
                isp->dmarx_dev.pre_frame = isp->dmarx_dev.cur_frame;
                isp->dmarx_dev.cur_frame.id = t.frame_id;
                isp->dmarx_dev.cur_frame.sof_timestamp = t.sof_timestamp;
-               isp->dmarx_dev.cur_frame.timestamp = t.frame_timestamp;
+               isp->dmarx_dev.cur_frame.timestamp =
t.sof_timestamp;//t.frame_timestamp;
                mode = t.mode;
                times = t.times;
                hw->cur_dev_id = id;
```

cif优化

cif主要耗时在于mipi传输buf时间，不同sensor不一样，打上下面cif驱动优化提交后可以直接通过预览时打印rkcif节点看具体耗时：

```
commit ec0640c0189029f10bc5f9e239f28ec9a0b206d2
Author: Zefa Chen <zefa.chen@rock-chips.com>
Date:   Mon Jul 5 17:39:39 2021 +0800

    media: rockchip: cif: mipi wakes up buf by line int

    Signed-off-by: Zefa Chen <zefa.chen@rock-chips.com>
    Change-Id: If10afeec22ce89a52f7c0e0e454005ca3c3cdc5e
    (cherry picked from commit 500585fdc15b6338a6157f967f29f6d01c98097e)
```

```
[root@RV1126_RV1109:/]# cat proc/rkcif*
Driver Version:v00.01.0a
Work Mode:ping pong
Monitor Mode:idle
aclk_cif:491519999
hclk_cif:245760000
dclk_cif:297000000
Input Info:
        src subdev:m01_f_sc500ai 1-0030
```

```
        interface:mipi csi2
        lanes:4
        vc channel: 0
        hdr mode: normal
        format:SBGGR10_1X10/2560x1440@30
        crop.bounds:(0, 0)/2560x1440
Output Info:
        format:BG10/2560x1440(0,0)
        compact:enable
        frame amount:296
        early:0 ms
        readout:16 ms
        rate:33 ms
        fps:30
        irq statistics:
                        total:296
                        csi over flow:0
                        csi bandwidth lack:0
                        all err count:0
                        frame dma end:296
```

可以看到耗时打印项：readout:16 ms，mipi传输耗时与sensor有关，一般是固定的，暂无法优化，所以cif驱动优化方案为：提前按行送数据给下一级模块（isp），打上上面cif优化补丁后，通过开机脚本配置line 数值，实测预览后画面是否正常。

测试方法可以是手在画面最下面晃动，观察底部画面行之间是否有图像撕裂现象，如果有撕裂(后级处理耗时接近或小于提前时间导致)，需要调大line数值，line配置命令如下：

```
diff --git a/oem/oem_uvcc/RkLunch.sh b/oem/oem_uvcc/RkLunch.sh
index 473c3a1..6c746d8 100755
--- a/oem/oem_uvcc/RkLunch.sh
+++ b/oem/oem_uvcc/RkLunch.sh
@@ -48,6 +48,15 @@ echo "camera_max_height= ${camera_max_height}"
 export CAMERA_MAX_WIDTH=${camera_max_width}
 export CAMERA_MAX_HEIGHT=${camera_max_height}

+#line config
+cif_line=$((${camera_max_height} / 4 * 3 ))
+echo $cif_line > /sys/devices/platform/rkcif_mipi_lvds/line_int_num
+
 #rockit log level ctrl: 1:fatal error; 2: error; 3: warning; 4:infomational;
5:debug level; 6:verbose
 export rt_log_level=3
```

isp/ispp优化

isp/ispp为耗时优化的重点，打开的模块越多，耗时越长，目前驱动中已默认在每一帧中带上时间戳信息，可以直接通过预览时打印proc/rkispp*等节点看具体耗时（delay打印）：

```
[root@RV1126_RV1109:/]# cat proc/rkispp*
rkispp0    Version:v01.06.00
clk_ispp   350000000
aclk_ispp  491519999
hclk_ispp  245760000
Interrupt  Cnt:22358 ErrCnt:0
```

```
Input      rkisp0 Format:FBC420 Size:2560x1440 (frame:5641 rate:32ms delay:10ms)
Output     rkispp_m_bypass Format:NV12 Size:2560x1440 (frame:5640 rate:32ms
delay:46ms)
TNR        ON(0xf00000f) (mode: 3to1) (global gain: disable) (frame:5641 time:9ms
idle) CNT:0x0 STATE:0x1e000000
NR         ON(0x17) (external gain: enable) (frame:5641 time:6ms working)
0x5f0:0x7002c 0x5f4:0x3e0b
SHARP      ON(0x1b) (YNR input filter: ON) (local ratio: ON) 0x630:0x72b
FEC        ON(0x80000021) (frame:5640 time:19ms idle) 0xc90:0x0
ORB        OFF(0x0)
Monitor    ON Cnt:0
```

另外查看isp和ispp状态可以直接查看/proc/rkisp*节点，会显示打开了哪些图像处理内部模块功能，由于图像效果直接影响这块耗时（噪点、hdr等），所以在延时优化之前，建议先完成图像turning 工作，图像效果稳定后再进行优化。

优化一：isp ispp 抬频，优化6ms+

```
diff --git a/drivers/media/platform/rockchip/isp/hw.c
b/drivers/media/platform/rockchip/isp/hw.c
index d6d3ca3..7b1a01e 100644
--- a/drivers/media/platform/rockchip/isp/hw.c
+++ b/drivers/media/platform/rockchip/isp/hw.c
@@ -353,13 +353,13 @@ static const struct isp_clk_info rv1126_isp_clk_rate[] = {
                .clk_rate = 20,
                .refer_data = 0,
        }, {
-               .clk_rate = 300,
+               .clk_rate = 600,
                .refer_data = 1920, //width
        }, {
-               .clk_rate = 400,
+               .clk_rate = 600,
                .refer_data = 2688,
        }, {
-               .clk_rate = 500,
+               .clk_rate = 600,
                .refer_data = 3072,
        }, {
                .clk_rate = 600,
diff --git a/drivers/media/platform/rockchip/ispp/hw.c
b/drivers/media/platform/rockchip/ispp/hw.c
index 95ca8f1..e0ac476e 100644
--- a/drivers/media/platform/rockchip/ispp/hw.c
+++ b/drivers/media/platform/rockchip/ispp/hw.c
@@ -181,13 +181,13 @@ static const struct ispp_clk_info rv1126_ispp_clk_rate[] =
{
                .clk_rate = 150,
                .refer_data = 0,
        }, {
-               .clk_rate = 250,
+               .clk_rate = 500,
                .refer_data = 1920 //width
        }, {
-               .clk_rate = 350,
+               .clk_rate = 500,
                .refer_data = 2688,
```

```
        }, {
-               .clk_rate = 400,
+               .clk_rate = 500,
                .refer_data = 3072,
        }, {
                .clk_rate = 500,
```

优化二：isp 添加line配置，提前送数据给tnr模块并行

打上下面提交补丁，默认line配置关闭，需要如CIF在脚本配置节点，不同sensor配置有差异，原理上line配置提前的时间需要小于ispp中tnr的处理时间，实测为准。

```
commit 35dd5b74d555d8b8f874549fe6ab76aadbbf5800
Author: Cai YiWei <cyw@rock-chips.com>
Date:   Thu Jul 8 18:16:32 2021 +0800

    media: rockchip: isp: frame buffer done early

    config wait-line to isp virtual device dts node,
    or echo value to debug node before open isp video.
    /sys/module/video_rkisp/parameters/wait_line

    Change-Id: I5c73c90117455663620b4c025e78aa6233ca40b9
    Signed-off-by: Cai YiWei <cyw@rock-chips.com>
```

注意：该line配置需要多留余量，若提前时间大于tnr耗时，可能会出现撕裂或卡住断流异常。

优化三：ispp 添加line配置，提前送数据给后级模块并行

打上下面提交补丁，默认line配置关闭，需要如cif/isp在脚本配置节点，不同sensor配置有差异，原理上line配置提前的时间需要小于后级处理时间，实测为准，若大于后级处理耗时会出现图像撕裂现象。

```
commit 4549cfa4134c1ee9d0ee2423a0e555486e5d5793
Author: Cai YiWei <cyw@rock-chips.com>
Date:   Wed Jul 7 09:27:45 2021 +0800

    media: rockchip: ispp: frame buffer done early

    config wait-line to ispp virtual device dts node,
    or ispp debug node before open ispp video.
    /sys/module/video_rkispp/parameters/wait_line

    for example: output is 2688x1520, config
    wait-line to 768 (128 align), vb2 buffer
    will done when poll image processing greater
    than 768, wait-line less than (height - 128) is valid.

    Change-Id: I4a448cc6baffbb5794eef91965e4b2bc349aa5ed
    Signed-off-by: Cai YiWei <cyw@rock-chips.com>
```

优化四：ispp tnr 3to1改2to1

通过rkispp0节点打印的信息可以确认TNR模块是否开启了3to1模式，该模式会在硬件处理中多延时一帧来合成图像，关闭则会提升一帧（33ms）延时收益，需要重点确认。关闭方法为IQ的xml文件中搜索3to1改成0配置即可，预览时通过rkispp0节点打印的信息确认是否已经是2to1.

参考line配置补丁：

下面为参考line配置补丁，可以先基于此配置再实测调优：

```
commit f855a3fb926555c3fed8b4d2b1a2323761455400
Author: Mark Huang <Mark Huang@rock-chips.com>
Date:   Thu Jul 29 09:36:38 2021 +0800

    oem_uvcc: add isp/ispp/cif line config

    Signed-off-by: Mark Huang <Mark Huang@rock-chips.com>

diff --git a/oem/oem_uvcc/RkLunch.sh b/oem/oem_uvcc/RkLunch.sh
index 473c3a1..6c746d8 100755
--- a/oem/oem_uvcc/RkLunch.sh
+++ b/oem/oem_uvcc/RkLunch.sh
@@ -48,6 +48,15 @@ echo "camera_max_height= ${camera_max_height}"
 export CAMERA_MAX_WIDTH=${camera_max_width}
 export CAMERA_MAX_HEIGHT=${camera_max_height}

+#line config
+isp_line=$((${camera_max_height} / 2))
+ispp_line=$((${camera_max_height} / 4 * 3 ))    #no fec
+cif_line=$((${camera_max_height} / 4 * 3 ))
+echo "isp_line= $isp_line , ispp_line= $isp_line, cif_line= $cif_line"
+echo  $isp_line> /sys/module/video_rkisp/parameters/wait_line
+echo $ispp_line > /sys/module/video_rkispp/parameters/wait_line
+echo $cif_line > /sys/devices/platform/rkcif_mipi_lvds/line_int_num
+
 #rockit log level ctrl: 1:fatal error; 2: error; 3: warning; 4:infomational;
5:debug level; 6:verbose
 export rt_log_level=3
```

isp/ispp部分优化也可以和直通优化方式结合，具体以客户产品实际测试效果和画质权衡为准。

rga处理和调度耗时优化

该部分和客户产品的应用处理有关系，默认SDK延时没有这一部分耗时，例子中该部分优化处理合并了EPTZ和rkrga插件的裁剪缩放处理部分，优化到5ms。

SDK中相关提交如下：

```
commit cb5eb4e8a2a57ea2229406675baa6900f659ba36
Author: lqh <kevin.lin@rock-chips.com>
Date:   Wed Jul 7 17:56:12 2021 +0800

    [aiserver/uvc_graph] Optimize UVC process.

    Using rkzoom node to operate ptz and small resolution display.
    Delete link that is no longer in use, now support 3 main link,
    under eptz mode, using eptz link, under nomoral display mode that
    resolution > 480P, using uvc mode, other sinces using uvc_zoom mode
    which support ptz feture, and 480P/240P 16:9 output.

    Signed-off-by: lqh <kevin.lin@rock-chips.com>
    Change-Id: Icf8fc50967f0be2de11e1fe6108bd5251fd92039

commit 9b7097526d0eebc7fdd4a779472338f994b70feb
```

```
Author: lqh <kevin.lin@rock-chips.com>
Date:   Tue Jul 6 14:10:04 2021 +0800

    [aiserver/vendor/zoom] add zoom operation.

    Using RTNodeVFilterZoom.cpp to instead operation in librockit.
    Combines eptz and ptz operation, reduce 1 time rga operation.

    Signed-off-by: lqh <kevin.lin@rock-chips.com>
    Change-Id: I3185f850b6b4eacff6807d04f690a161ea9c5d81
```

mpp编码耗时优化

这部分耗时和编码格式及分辨率大小有正相关，编码耗时：mjpeg>h264≈h265 。

1440p分辨率为例，各格式编码部分耗时情况：

编码格式	耗时 （ms）
MJPEG	25
H264	16
H265	16

例子中针对1440p mjpeg场景使用了单独的分帧编码优化方案，需要根据具体各个分辨率格式配置，需要单独mpp库支持，不通用，仅介绍通用优化方案：vpu抬频。

```
diff --git a/arch/arm/boot/dts/rv1126.dtsi b/arch/arm/boot/dts/rv1126.dtsi
index 4b0024c..6e7c2b2 100644
--- a/arch/arm/boot/dts/rv1126.dtsi
+++ b/arch/arm/boot/dts/rv1126.dtsi
@@ -2145,7 +2145,7 @@
                interrupts = <GIC_SPI 74 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&cru ACLK_JPEG>, <&cru HCLK_JPEG>;
                clock-names = "aclk_vcodec", "hclk_vcodec";
-               rockchip,normal-rates = <400000000>, <0>;
+               rockchip,normal-rates = <500000000>, <0>;
                rockchip,advanced-rates = <500000000>, <0>;
                rockchip,default-max-load = <2088960>;
                resets = <&cru SRST_JPEG_A>, <&cru SRST_JPEG_H>;
```

注意：非参考sdk公版硬件设计，需要注意vpu一路电压是否不够，若不够一般会出现画面部分花点现象。

usb处理和调度耗时优化

例子中针对1440p mjpeg场景配合编码同步使用了单独的分帧传输优化方案，需要根据具体各个分辨率格式配置，需要单独编码补丁包支持，不通用，仅介绍通用优化方案：优化传输过程中cpu idle引入的调度等待耗时,优化约8ms。

```
kernel$ git show c1b9c11178c08d3470558e0a815184342594622b
commit c1b9c11178c08d3470558e0a815184342594622b
Author: William Wu <william.wu@rock-chips.com>
Date:   Fri May 7 11:49:41 2021 +0800
```

```
    usb: gadget: f_uvc: disallow the CPU to enter deeper sleep when streamon

    I test on RK3399 IND EVB running Android R system, and
    set the USB gadget as UVC function via configfs, both
    the USB 2.0 and 3.0 UVC have preview abnormal problems.
    If the CPU enter deeper sleep, the USB DMA transfer gets
    corrupted, and the UVC ISOC transmission lost data.

    This patch puts in a "pm_qos_latency" requirement which
    will keep the CPU out of the deeper sleep states when
    UVC stream on. And allow the CPU to enter deeper sleep
    state after UVC stream off.

    Change-Id: I808290f124c6a32da3888819348093a205bfad61
    Signed-off-by: William Wu <william.wu@rock-chips.com>

diff --git a/drivers/usb/gadget/function/f_uvc.c
b/drivers/usb/gadget/function/f_uvc.c
index 9f399f4..4998b96 100644
--- a/drivers/usb/gadget/function/f_uvc.c
+++ b/drivers/usb/gadget/function/f_uvc.c
@@ -1163,6 +1163,7 @@ static struct usb_function_instance *uvc_alloc_inst(void)
        opts->streaming_interval = 1;
        opts->streaming_maxpacket = 1024;
        opts->uvc_num_request = UVC_NUM_REQUESTS;
+       opts->pm_qos_latency = 0;

        ret = uvcg_attach_configfs(opts);
        if (ret < 0) {
diff --git a/drivers/usb/gadget/function/u_uvc.h
b/drivers/usb/gadget/function/u_uvc.h
index be26665..cfa81bc 100644
--- a/drivers/usb/gadget/function/u_uvc.h
+++ b/drivers/usb/gadget/function/u_uvc.h
@@ -87,6 +87,7 @@ struct f_uvc_opts {
         */
        struct mutex                    lock;
        int                             refcnt;
+       int                             pm_qos_latency;
 };

 #endif /* U_UVC_H */
diff --git a/drivers/usb/gadget/function/uvc.h
b/drivers/usb/gadget/function/uvc.h
index 82ad867..eed4285 100644
--- a/drivers/usb/gadget/function/uvc.h
+++ b/drivers/usb/gadget/function/uvc.h
@@ -14,6 +14,7 @@
 #include <linux/spinlock.h>
 #include <linux/usb/composite.h>
 #include <linux/videodev2.h>
+#include <linux/pm_qos.h>

 #include <media/v4l2-device.h>
 #include <media/v4l2-dev.h>
@@ -117,6 +118,8 @@ struct uvc_device {
        enum uvc_state state;
        struct usb_function func;
```

```
        struct uvc_video video;
+       /* for creating and issuing QoS requests */
+       struct pm_qos_request pm_qos;

        /* Descriptors */
        struct {
diff --git a/drivers/usb/gadget/function/uvc_configfs.c
b/drivers/usb/gadget/function/uvc_configfs.c
index 4da4a34..4233770 100644
--- a/drivers/usb/gadget/function/uvc_configfs.c
+++ b/drivers/usb/gadget/function/uvc_configfs.c
@@ -2775,6 +2775,7 @@ UVCG_OPTS_ATTR(streaming_interval, streaming_interval,
16);
 UVCG_OPTS_ATTR(streaming_maxpacket, streaming_maxpacket, 3072);
 UVCG_OPTS_ATTR(streaming_maxburst, streaming_maxburst, 15);
 UVCG_OPTS_ATTR(uvc_num_request, uvc_num_request, UVC_MAX_NUM_REQUESTS);
+UVCG_OPTS_ATTR(pm_qos_latency, pm_qos_latency, PM_QOS_LATENCY_ANY);

 #undef UVCG_OPTS_ATTR

@@ -2839,6 +2840,7 @@ static struct configfs_attribute *uvc_attrs[] = {
        &f_uvc_opts_attr_streaming_maxburst,
        &f_uvc_opts_attr_uvc_num_request,
        &f_uvc_opts_attr_device_name,
+       &f_uvc_opts_attr_pm_qos_latency,
        NULL,
 };

diff --git a/drivers/usb/gadget/function/uvc_video.c
b/drivers/usb/gadget/function/uvc_video.c
index f1db6a3..ae95de3 100644
--- a/drivers/usb/gadget/function/uvc_video.c
+++ b/drivers/usb/gadget/function/uvc_video.c
@@ -13,6 +13,7 @@
 #include <linux/usb/gadget.h>
 #include <linux/usb/video.h>
 #include <linux/uvcinfo.h>
+#include <linux/pm_qos.h>

 #include <media/v4l2-dev.h>

@@ -346,9 +347,13 @@ int uvcg_video_enable(struct uvc_video *video, int enable)

                uvc_video_free_requests(video);
                uvcg_queue_enable(&video->queue, 0);
+               if (pm_qos_request_active(&uvc->pm_qos))
+                       pm_qos_remove_request(&uvc->pm_qos);
                return 0;
        }

+       pm_qos_add_request(&uvc->pm_qos, PM_QOS_CPU_DMA_LATENCY,
+                          opts->pm_qos_latency);
        if ((ret = uvcg_queue_enable(&video->queue, 1)) < 0)
                return ret;

diff --git a/drivers/usb/gadget/legacy/webcam.c
b/drivers/usb/gadget/legacy/webcam.c
index 89e7210..e68d921 100644
```

```
--- a/drivers/usb/gadget/legacy/webcam.c
+++ b/drivers/usb/gadget/legacy/webcam.c
@@ -383,6 +383,7 @@ webcam_bind(struct usb_composite_dev *cdev)
        uvc_opts->hs_streaming = uvc_hs_streaming_cls;
        uvc_opts->ss_streaming = uvc_ss_streaming_cls;
        uvc_opts->uvc_num_request = UVC_NUM_REQUESTS;
+       uvc_opts->pm_qos_latency = 0;

        /* Allocate string descriptor numbers ... note that string contents
         * can be overridden by the composite_dev glue.
```