USB Performance Analysis Guide

发布版本：1.1

作者邮箱：wulf@rock-chips.com

日期：2019-01-09

文档密级：公开资料

前言

概述

本文档提供Rockchip平台USB 2.0/3.0模块传输性能分析的方法以及支持的最大传输速率。

本文档将分别对USB Host和USB Device(Peripheral)两部分进行介绍。其中，USB Host将会介绍USB Disk、USB Ethernet以及USB Camera的传输性能分析。USB Device(Peripheral)将会介绍USB MTP、USB MSC(Mass Storage Class)、USB Gadget Webcam、USB Rndis、USB Gadget HID的传输性能分析。

产品版本

芯片名称	内核版本
30系列、31系列、32系列、33系列、PX系列、Sofia、RV1108	Linux3.10和Linux4.4

读者对象 本文档（本指南）主要适用于以下工程师： 硬件工程师 软件工程师 技术支持工程师

修订记录

日期	版本	作者	修改说明
2017-12-25	V1.0	吴良峰	初始版本
2019-01-09	V1.1	吴良峰	使用markdownlint修订格式

---

1 USB 理论传输速率分析

1.1 USB Communication Flow

USB的通信流模型如图1-1所示，采用分层的结构，一台主机与一个USB设备间的连接是由许多层上的连接组成 。USB总线接口层提供了主机和设备之间的物理连接、发送和接收数据。USB设备层对USB系统软件是可见的，系统软件基于它所见的设备层来完成对设备的一般的USB操作。USB应用层可以通过与之相配合的客户软件向主机提供一些额外的功能。USB设备层和应用层的通信是逻辑上的，对应于这些逻辑通信的实际物理通信由USB总线接口层来完成。

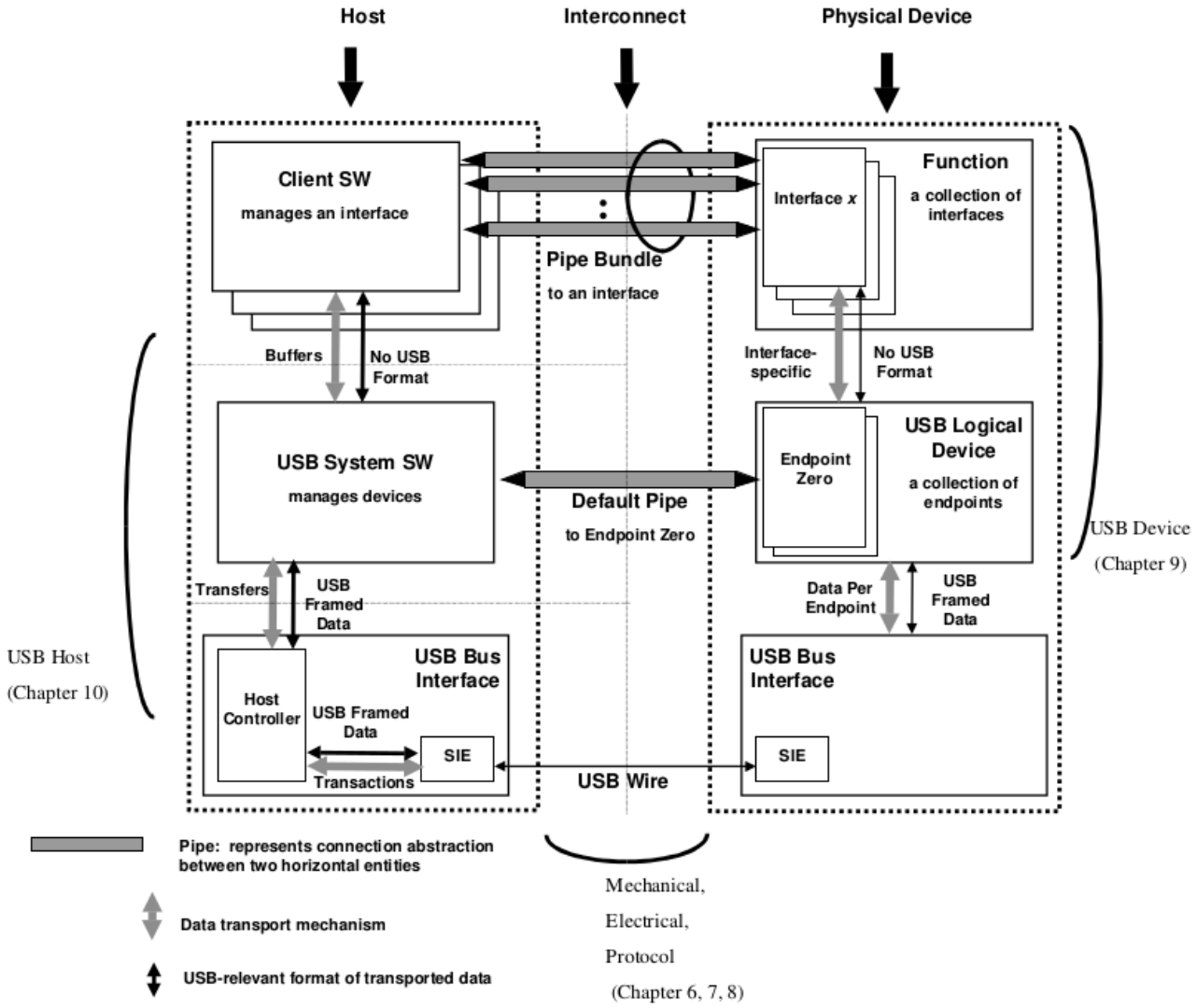

图1-1 USB Communication Flow

1.2 USB 物理层传输速率

USB物理层的传输速率是指USB物理总线的传输速率。USB总线目前支持USB1.0、USB1.1、USB2.0、USB3.0（USB3.1 Gen1）、USB 3.1 Gen2五种标准传输速率，如下表1-1所示。

表1-1 USB物理层最大传输速率

USB版本	最大传输速率	速率称号
USB 1.0	1.5Mbps	低速（Low-Speed）
USB 1.1	12Mbps	全速（Full-Speed）
USB 2.0	480Mbps	高速（High-Speed）
USB 3.0（USB 3.1 Gen1）	5Gbps	超速（Super-Speed）
USB 3.1 Gen2	10Gbps	暂未定义（Super-Speed+）

1.3 USB 协议层传输速率

不同的USB设备类型，采用不同的USB设备类协议，比如：USB Disk采用的是USB Mass Storage Class规范的子类规范（USB Mass Storage Class Bulk-Only Transport），USB Camera 采用的是USB Video Class规范，USB HID采用的是USB Human Interface Devices Class规范，更多的规范，请在USB-IF官网查看。

不同的USB设备类协议，定义了不同的数据包格式，有些设备类协议还需要命令包和握手包，并且，不同的USB设备类协议采用了不同的USB传输类型，比如：USB Disk采用批量传输类型（Bulk Transfers），USB Camera采用同步传输类型（Isochronous Transfers），USB HID采用中断传输类型（Interrupt Transfers）。

USB体系支持四种传输类型：

- 控制传输（Control Transfers）

  主要用于在设备连接时对设备进行枚举以及其他因设备而异的特定操作；
- 中断传输（Interrupt Transfers）

  用于对延迟要求严格、小量数据的可靠传输，如键盘、游戏手柄等HID设备；
- 批量传输（Bulk Transfers）

  用于对延迟要求宽松，大量数据的可靠传输，如U盘、USB以太网卡等；
- 同步传输（Isochronous Transfers）

  用于对可靠性要求不高的实时数据传输，如USB Camera、USB Audio；

不同的USB传输类型，在USB物理总线上并没有太大的区别，只是在传输机制、主机安排传输任务、可占用USB带宽的限制以及最大包长度有一定的差异，如下表1-2所示。

表1-2 USB的四种传输类型

传输类型	**USB 2.0**	**USB 3.0**
控制传输	Max Packet： 64Bytes Burst：不支持 仅支持OUT的流控制（PING）	Max Packet： 512Bytes Burst：1 支持流控制（ERDY）
中断传输	Max Packet： 0 ～ 1024 ×3 Bytes Max Transfer Speed：23.44MBps Burst：不支持 不支持流控制	Max Packet： 0 ～ 1024 ×3 Bytes Max Transfer Speed：23.44MBps Burst：1 ~ 3 支持流控制（ERDY）
批量传输	Max Packet： 512Bytes Max Transfer Speed：53.25MBps Burst：不支持 仅支持OUT的流控制（PING）	Max Packet： 1024Bytes Max Transfer Speed：400MBps？ Burst：1 ～ 16 支持流控制（ERDY））
同步传输	Max Packet： 0 ～ 1024 ×3 Bytes Max Transfer Speed：23.44MBps Burst：不支持 不支持流控制	Max Packet： 0 ～ 1024 ×16 × 3Bytes Max Transfer Speed：375.04MBps Burst：1 ~ 16 支持流控制（PING -> PING_RESPONSE）

由表格1-2可以看出，不同的USB传输类型，最大的理论传输速率也是不同的。并且，由于受到USB传输类型的最大速率限制以及USB设备类协议的交互影响，USB的实际传输速率远小于USB物理总线上的传输速率。

1.4 影响USB传输速率的主要因素

USB的传输速率主要受如下几方面的影响：

1. **USB信号质量**

   USB信号质量可以通过测试USB眼图来评估，如果USB信号质量差，容易导致USB物理总线丢包、重传，影响USB传输的速率。

2. **USB 控制器的AHB CLK和DMA Burst Length**

   更高的AHB CLK和更大的DMA Burst Length，可以提高USB控制器的DMA传输效率。

3. **CPU运行频率的影响**

   CPU运行频率会影响代码的执行效率和数据拷贝memcpy的效率，所以提高CPU频率，可以提高USB驱动和应用层的执行效率，从而提高USB传输速率。

4. **USB设备类驱动提交的URB (USB Request Block)的buffer大小**

   提交的URB buffer大小，也即请求USB控制器一次传输的数据块大小。对于USB MSC（Mass Storage Class）类设备，如USB Disk，USB MTP，如果不考虑数据块大小对存储颗粒（EMMC或NAND）的读写性能的影响，可以尽量增加buffer的大小，以降低USB控制器传输的中断数量。Rockchip的USB DWC2控制器支持一次最大传输512KB，USB DWC3控制器支持一次最大传输16MB - 1 Bytes。

5. **存储介质读写性能的影响**

   如果是USB MSC（Mass Storage Class）类设备，如USB Disk的读写速率，受USB Host端存储介质以及U盘本身存储介质的读写性能影响.

6. **文件系统格式的影响**

U盘拷贝速率容易受文件系统的影响，常见的文件系统格式包括：VFAT、EXT4和NTFS。对于VFAT/EXT4两种文件系统格式的传输机制，kernel的block层会自动将小的数据块merge为120K，再写入磁盘。而NTFS的写入磁盘操作是在用户空间，数据块不会由block层merge。所以，如果应用层每次请求的数据块太小（如4KB），NTFS文件系统格式的U盘，拷贝速率一般会明显慢于VFAT/EXT4的文件系统格式。详细的分析，请参考2.1 USB Disk传输速率分析

2 USB Host传输性能分析

2.1 USB Disk传输速率分析

测试方法：

假设USB Disk插入USB Host接口后，枚举的信息如下：

磁盘分区为：

`/dev/block/sda1`

挂载路径为：

`/mnt/media_rw/0E64-5F76/`

USB Host EMMC文件存储路径：`/sdcard/.`

下面的方法，是基于上述的假设进行说明。

Note:

- **test文件，应该足够大**。比如，在3399 EVB 4G DDR上测试，建议test文件为2G，这样才能保证数据从缓存中刷入到磁盘，实测如果使用1024MB文件，速率是不准确的。原因是`proc/sys/vm/dirty_background_ratio`一般为5%，也即4G内存，文件系统缓存脏页数量要达到4G×5% = 204.8MB，系统才会将缓存的脏页异步地刷入存储介质。
- dd命令，可以加**conv=fsync**，表示将缓存中的数据写入磁盘。

USB Disk的传输速率测试，通常有以下两种方法：

- **方法1：手动或者使用cp命令拷贝大文件（如电影），然后计时，统计速率；**

  step1. 使用dd命令创建一个test文件，用于step2的拷贝测试使用

  `busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=64K count=16K`

  step2. 使用cp命令统计拷贝时间

  `echo 3 > /proc/sys/vm/drop_caches` `time cp /mnt/media_rw/0E64-5F76/test /sdcard/.`

  上述方法，是测试USB Disk的读速率，如果需要测试USB Disk的写速率，只要修改读写路径即可。

- **方法2：使用dd命令测试**

  测试USB Disk的读速率方法：

  step1. 使用dd命令创建一个test文件，用于后续的拷贝测试使用

  `busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=64K count=16K`

  step2.

  每次执行dd命令前，先清缓存

  `echo 3 > /proc/sys/vm/drop_caches`

测试4K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=4K count=256K`

step3.

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试64K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=64K count=16K`

step4.

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试128K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=128K count=8K`

step5.

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试512K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=512K count=2K`

测试USB Disk的写速率方法：

step1.

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试4K数据块的写速率

`busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=4K count=256K`

step2.

`rm of=/mnt/media_rw/0E64-5F76/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试64K数据块的写速率

`busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=64K count=16K`

step3.

`rm of=/mnt/media_rw/0E64-5F76/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试128K数据块的写速率

`busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=128K count=8K`

step4.

`rm of=/mnt/media_rw/0E64-5F76/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试512K数据块的写速率

`busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=512K count=2K`

Note：

- 方法1可以统计USB Disk的实际数据读写速率，但测试结果同时受Host端存储介质、文件系统性能和USB控制器传输速率的影响，无法准确说明USB控制器的传输性能。
- 方法2可以排除Host端存储介质和文件系统的影响，准确统计USB控制器的读写U盘的传输性能。但需要注意的是，USB Disk（Device端）的文件系统仍然会影响拷贝，如果要排除USB Disk文件系统的影响，需要直接读写/dev/路径下的sd*分区节点，但同时会破坏USB Disk原有的数据，所以要慎用。
- dd命令，可以加**conv=fsync**，表示将缓存中的数据写入磁盘，这样可以完全排除缓存的影响。当然，如果测试的文件足够大，缓存的影响也是比较小的。
- 测试时，建议两种方法都使用。先使用方法1测试，如果测试结果无法达到预期的传输速率，则进一步使用方法2分析是否传输瓶颈在USB控制器。详细的分析方法，将在"测试结果分析"章节中说明。

测试结果：

- **USB 2.0 Host**

  使用dd命令测试，Rockchip USB 2.0 Host接口的USB Disk读写速率通常在 **25MBps** ～ **35MBps**。

- **USB 3.0 Host**

  使用dd命令测试，Rockchip USB 3.0 Host接口的USB3 Disk（不支持UAS）读写速率通常在 **60MBps** ～ **100MBps**。 支持**UAS**的USB3 Disk，最大读写速率约为**350MBps**。

Note：

- 不同的USB3 Disk，读写性可能有明显差异，并且，有些USB3 Disk的读速率比写速率高3～5倍。所以，评估USB控制器的传输速率，应该尽量多测试几种不同型号的USB3 Disk，并使用同样的测试方法，在PC Ubuntu测试同样USB3 Disk的读写速率，以作对比参考。
- UAS的支持：RK3399/RK3328 USB3 控制器支持UASP（USB Attached SCSI PROTOCOL），该功能可以大大提高USB Disk的读写速率。如果要使用UAS功能，首先，内核需要enable CONFIG_USB_UAS（kernel3.10和kernel4.4默认都已经enable），其次，USB3 Disk需要支持UAS功能。
- 可以从枚举的log中，确认是否使用UAS协议

```
[80373.513866] usb 4-1: new SuperSpeed USB device number 2 using xhci_hcd
[80373.534854] usb 4-1: New USB device found, idVendor=174c, idProduct=55aa
[80373.534871] usb 4-1: New USB device strings: Mfr=2, Product=3, SerialNumber=1
[80373.534873] usb 4-1: Product: CHIPFANCIER
[80373.534874] usb 4-1: Manufacturer: WINTOGO
[80373.534876] usb 4-1: SerialNumber: 000001485761816
[80373.537036] scsi host6: uas （说明将以UAS协议进行通信）
[80373.537648] scsi 6:0:0:0: Direct-Access     WINTOGO  CHIPFANCIER      0    PQ: 0
ANSI: 6
```

测试结果分析与性能优化方向：

实例：

本文档以RK3399 EVB 读写USB3 Disk的典型应用场景作分析，其他应用场景可以参考下面的方法。

测试环境：RK3399 EVB （2G DDR + 16G EMMC）

USB3 Disk型号为 SanDisk Type-C USB3.0 32G （NTFS 文件系统）

测试场景：测试从USB3 Disk拷贝大文件到3399 EMMC的速率

测试命令： cp，dd

假设USB3 Disk挂载路径：/mnt/media_rw/0E64-5F76/

EMMC文件存储路径：/sdcard/.

Note：

如果测试机器的DDR为4G，建议test文件为**2GB**，以避免缓存的影响。此外，dd命令，可以加**conv=fsync**，表示将缓存中的数据写入磁盘，这样可以完全排除缓存的影响。当然，如果测试的文件足够大，缓存的影响也是比较小的。

step1. 使用**dd**命令创建一个**test**文件（**1GB**大小），用于后续的拷贝测试使用

`busybox dd if=/dev/zero of=/mnt/media_rw/0E64-5F76/test bs=64K count=16K`

step2. 使用**cp**命令测试测试从**USB3 Disk**拷贝大文件到**3399 EMMC**的速率

`time cp /mnt/media_rw/0E64-5F76/test /sdcard/.` (test为1GB文件)

测试结果： 0m48.42s real 0m00.29s user 0m18.17s system

拷贝速率为：1024M/48 = 21MBps 该速率明显慢于USB3的正常拷贝速率

step3. 直接从**/dev/block/U**盘分区 拷贝到**EMMC**，以排查**3399**文件系统的影响

`time cp /dev/block/sda1 /sdcard/test` （该命令可以绕过文件系统直接拷贝）

拷贝速率约为： 20MBps

说明拷贝速率慢与3399文件系统本身没有关系。

step4. 分别使用**dd**命令测试**U**盘和**3399 EMMC**的读写性能

- 测试**USB3 Disk**读性能

  每次执行dd命令前，先清缓存

  `echo 3 > /proc/sys/vm/drop_caches`

  测试4K数据块的读速率

  `busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=4K count=256K`

  结果：1073741824 bytes (1.0GB) copied, 14.183164 seconds, 72.2MB/s

  每次执行dd命令前，先清缓存

  `echo 3 > /proc/sys/vm/drop_caches`

  测试64K数据块的读速率

  `busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=64K count=16K`

  结果：1073741824 bytes (1.0GB) copied, 12.546069 seconds, 81.6MB/s

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试128K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=128K count=8K`

结果：1073741824 bytes (1.0GB) copied, 12.601817 seconds, 81.3MB/s

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试512K数据块的读速率

`busybox dd if=/mnt/media_rw/0E64-5F76/test of=/dev/null bs=512K count=2K`

结果：1073741824 bytes (1.0GB) copied, 12.402147 seconds, 82.6MB/s

- 测试**3399 EMMC**写性能

每次执行dd命令前，先清缓存

`echo 3 > /proc/sys/vm/drop_caches`

测试4K数据块的写速率

`busybox dd if=/dev/zero of=/sdcard/test bs=4K count=256K`

结果：1073741824 bytes (1.0GB) copied, 39.525297 seconds, 25.9MB/s

`rm of=/sdcard/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试64K数据块的写速率

`busybox dd if=/dev/zero of=/sdcard/test bs=64K count=16K`

结果： 1073741824 bytes (1.0GB) copied, 37.505206 seconds, 27.3MB/s

`rm of=/sdcard/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试128K数据块的写速率

`busybox dd if=/dev/zero of=/sdcard/test bs=128K count=8K`

结果： 1073741824 bytes (1.0GB) copied, 37.461603 seconds, 27.3MB/s

`rm of=/sdcard/test`

`sync`

`echo 3 > /proc/sys/vm/drop_caches`

测试512K数据块的写速率

`busybox dd if=/dev/zero of=/sdcard/test bs=512K count=2K`

结果：1073741824 bytes (1.0GB) copied, 37.086531 seconds, 27.6MB/s

从上述的测试结果看，USB3 Disk的读性能很好，小块和大块读性能都基本一致。而EMMC的写性能明显都很差。从USB3 Disk拷贝大文件到3399 EMMC的传输速率瓶颈在3399的EMMC颗粒。

上面分析的EMMC颗粒，是4K，64K，128K，512K数据块的写都比较慢，然而，在Rockchip平台上，大部分的EMMC颗粒可能只有4K的写速率比较慢（包括USB3 Disk也如此），所以，如果测试的EMMC写速率只有4K比较慢，那需要进一步分析是否与4K有关系。

step5. 分析**cp**命令的行为和**block**层的行为

分析工具： **strace**，**blktrace**

- 使用**strace**命令跟踪**cp**行为

  `strace cp /mnt/media_rw/0E64-5F76/test /sdcard/.`

  结果表明：RK3399平台的cp命令，读写的数据块都是4KB

- 使用**blktrace**命令分析**block**层的**merge**行为

  几个与拷贝相关的重要参数：

  `max_hw_sectors_kb`

  `max_sectors_kb`

  max_hw_sectors_kb和max_sectors_kb控制的是block层执行merge操作后写入存储介质的数据块大小，U盘一般是120KB，EMMC一般是512KB。

  EMMC：

  ```
  /sys/block/mmcblk1/queue/max_hw_sectors_kb
  /sys/block/mmcblk1/queue/max_sectors_kb
  ```

  USB Disk：

  ```
  /sys/block/sda/queue/max_hw_sectors_kb
  /sys/block/sda/queue/max_sectors_kb
  ```

  其中，USB Disk的max_hw_sectors_kb设置在代码：

  ```
  drivers/usb/storage/scsiglue.c
  /*limit the total size of a transfer to 120 KB */
  .max_sectors =  240, （单位为block512 bytes）
  ```

  该参数可以由上层配置，如RK3399 Type-C1 口：

  `/sys/devices/platform/usb@fe900000/fe900000.dwc3/xhci-hcd.9.auto/usb4/4-1/4-1:1.0/host0/target0:0:0/0:0:0:0/max_sectors`

  查看io调度器：

  `rk3399:/data# cat /sys/block/mmcblk1/queue/scheduler`

`noop [cfq]`（使用cfq，表明理论上会执行merge行为，将小数据块进行merge）

blktrace为Android自带的工具，但默认可能没有支持，需要先修改blktrace文件夹下的mk文件，并编译该工具，然后使用adb push到/system/bin。

修改Android.mk文件： 将文件第一行的`BUILD_BLKTRACE:= false`改为`BUILD_BLKTRACE:= true`

要使用blktrace，内核需要增加CONFIG_BLK_DEV_IO_TRACE，如下：

```
Kernel hacking  --->
    [*]Tracers  --->
        [*]  Support for tracing block IO actions
```

blktrace的使用方法参考：

抓blktrace的数据：

1. 创建文件夹用于保存数据

   `mkdir /dev/blktrace`

2. 启动blktrace，开始抓数据

   `blktrace -d /dev/block/mmcxxx -o /dev/blktrace/blktrace &`

   (其中，/dev/block/mmcxxx为要分析的block，如3399的mmcblk1)

3. 执行拷贝命令

   `busybox dd if=/dev/zero of=/data/test bs=4K count=256K`（执行拷贝）

4. 停止blktrace

   拷贝完成后，kill blktrace的线程，blktrace的数据就会自动保存在/dev/blktrace路径下，也可以在blktrace后面加-w参数，表示trace多少秒后，自动关闭blktrace

分析blktrace的数据：

`blkparse –i /dev/blktrace/ blktrace.blktrace.0 –o /dev/blktrace/blkparse.txt`

根据blktrace的统计结果，cfq会将4k的小块数据合并为1024个扇区，即512K，才写到EMMC，如下图2-1所示。

```
179,0    1     7832     4.947706748     6  G   W 18996840 + 1024 [kworker/u12:0]
179,0    0     2410     4.952877415   168  C   W 18977384 + 1024 [0]
179,0    0        0     4.953095582     0  m   N cfq6A  complete rqnoidle 0
179,0    0        0     4.953101415     0  m   N cfq6A  set_slice=12
179,0    0        0     4.953115707     0  m   N cfq6A  dispatch_insert
179,0    0        0     4.953121540     0  m   N cfq6A  dispatched a request
179,0    0        0     4.953124165     0  m   N cfq6A  activate rq, drv=2
179,0    0     2411     4.953125623   168  D   W 18979432 + 1024 [mmcqd/1]
179,0    1     7833     4.955261498     6  A   W 18997864 + 1024 <- (179,15) 15064616
179,0    1     7834     4.955263832     6  Q   W 18997864 + 1024 [kworker/u12:0]
179,0    1     7835     4.955299415     6  G   W 18997864 + 1024 [kworker/u12:0]
```

图2-1 blktrace data

说明3399的block层的merge行为是没有问题的。

更详细的blktrace使用方法，请参考blktrace分析IO

step5. 分析**USB Disk**文件系统格式的影响

USB Disk常用的文件系统格式包括：VFAT、EXT4和NTFS。对于VFAT/EXT4两种文件系统格式的传输机制，kernel的block层会自动将小的数据块merge为大数据块，再写入磁盘。而NTFS是基于fuse，写入磁盘操作是在用户空间，并且会多了拷贝操作，所以，如果应用层每次请求的数据块太小（如4KB），对于NTFS文件系统格式的USB Disk，在CPU变频的情况下，拷贝速率可能会低于VFAT/EXT4文件系统格式。比如，在RK3399 Android平台，cp命令每次请求的数据块就只有4KB，所以，在RK3399 Android平台，NTFS格式的USB3 Disk读写速率往往会比较差。

优化方向：

根据上面的分析，影响USB Disk拷贝的主要因素，有如下几个：

- USB Disk的读写性能；
- USB Host端的存储颗粒读写性能；
- USB Host控制器传输性能；
- 文件系统格式；
- 应用层一次请求的数据块大小；
- CPU运行频率；

可以参考本文档提供的方法，先定位传输瓶颈，再进行优化。

2.2 USB Ethernet传输速率分析

测试方法：

测试工具：iperf

iperf命令是一个网络性能测试工具。iperf可以测试TCP和UDP带宽质量。iperf可以测量最大TCP带宽，具有多种参数和UDP特性。iperf可以报告带宽，延迟抖动和数据包丢失。

- iperf的官方网址：

  https://iperf.fr/

- iperf下载地址：

  支持Windows /Android/Ubuntu/macOS等系统，下载地址

  https://iperf.fr/iperf-download.php

  此外，Ubuntu系统可以直接使用如下命令安装iperf

  `sudo apt install iperf`

  Andorid系统可以安装iperf.apk

- iperf的使用方法：

  https://iperf.fr/iperf-doc.php

  http://man.linuxde.net/iperf

带宽测试通常采用**UDP模式**，因为能测出极限带宽、时延抖动、丢包率。在进行测试时，首先以链路理论带宽作为数据发送速率进行测试，例如，从客户端到服务器之间的链路的理论带宽为100Mbps，先用`-b 100M`进行测试，然后根据测试结果（包括实际带宽，时延抖动和丢包率），再以实际带宽作为数据发送速率进行测试，会发现时延抖动和丢包率比第一次好很多，重复测试几次，就能得出稳定的实际带宽。

实例

UDP模式

PCUbuntu作服务器端，RK3399的USB2/USB3 Host接口连接UGREEN USB3 Ethernet，然后通过以太网线连接到PC的以太网口，形成完整的测试链路。

PC Ubuntu服务器端：

首先，配置ipv4地址、子网掩码和网关，如下图2-2

图2-2 PC Ubuntu ip地址配置

然后，执行如下命令：

```
wlf@wlf-ubuntu:~$ iperf -u -s
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

3399 客户端：

首先，查看USB以太网卡的信息

```
eth1      Link encap:Ethernet  HWaddr 60:38:E0:E3:20:96
          inet6 addr: fe80::6238:e0ff:fee3:2096/64 Scope: Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:168
```

然后，配置ip地址

`ifconfig eth1 192.168.1.2 up`

再执行如下命令，开始测试带宽

```
iperf -u -c 192.168.1.1 -b 1000M -t 60
```

Note：-b 表示UDP模式使用的带宽，单位bits/sec。如果是USB2.0网卡，建议设置为-b 100M，如果是USB3.0网卡，建议设置为-b 1000M。

测试结果：

使用UGREEN USB3.0 Ethernet，实际的测试带宽如下：

```
------------------------------------------------------------
Client connecting to 192.168.1.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 40116 connected with 192.168.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-60.0 sec  5.61 GBytes   803 Mbits/sec
[  3] Sent 4096616 datagrams
```

使用同样的测试方法，测试CISCO USB2.0 Ethernet的带宽约为95Mbits/sec

测试结果分析：

Rockchip平台USB Ethernet的正常传输速率如下：

USB2.0 Host：上行和下行均为**95 ～ 100 Mbits/sec**

USB3.0 Host：上行和下行均为**750 ～ 800 Mbits/sec**

如果实际产品测试无法达到上述速率，请先使用ifconfig和tcpdump工具，排查是否误码率太高，如果误码率太高，请测试USB眼图，确认USB的信号质量是否符合要求。

2.3 USB Camera传输速率分析

测试方法：

使用系统自带的Camera APK进行图像预览，并通过串口执行logcat，查看打印的实时帧率，如下：

`CameraHal: debugShowFPS(622): Camera 1312 Frames, 30.000 FPS`

uvc驱动调试方法：

打印uvc trace信息

```
echo 0xffff > /sys/module/uvcvideo/parameters/trace
echo 8 > /proc/sysrq-trigger
```

关闭uvc trace

`ehco 0 > /sys/module/uvcvideo/parameters/trace`

查看uvc decode统计信息

cat /d/usb/uvcvideo/*/stats

测试结果：

Rockchip平台USB Camera支持的最大帧率如下：

表2-1 Rockchip平台USB Camera最大帧率

	YUYV格式	**MJPEG格式**	**H264格式**
USB 2.0	640×480 @30fps	1280×720 @30fps	1920×1080 @30fps
USB 3.0	640×480 @60fps	1280×720 @60fps	1920×1080 @60fps

Note：上述帧率的测试前提条件是每个USB控制器只连接一个USB Camera ， 如果同时连接两个或多个USB Camera，要以实测为准。

测试结果分析与性能优化：

大部分的USB Camera都是采用同步传输类型进行USB通信，USB2.0的同步传输理论最大带宽是23.44MBps，USB3.0的同步传输理论最大带宽是375.04MBps。USB Camera的最大帧率一般受如下两个方面的影响：

a). USB Camera实际能够输出的最大帧率；

b). USB Host驱动的同步传输性能；

对于因素a)，可以在PC上验证USB Camera的性能。对于因素b)，可以考虑提高USB QOS的优先级、提高USB中断的响应速度、提高USB控制器的AHB CLK以及DMA burst length。

3 USB Device传输性能分析

3.1 USB MTP传输速率分析

测试方法：

将待测设备连接到PC，设置USB为MTP模式，然后手动拷贝大文件（如1GB以上的电影），用秒表计时，再计算平均传输速率。

测试时注意以下几点：

- 待测设备应保持在静态桌面，避免进休眠；
- 待测设备连接到PC后，需要等待媒体库扫描完成（可以观察logcat打印），再执行拷贝操作；

测试结果：

实例

测试平台：RK3399 BOX SDK Android N + Kernel 4.4

测试说明：

Read：RK3399 -> PC

Write: PC -> RK3399

16K，64K，1M表示MTP驱动的Tx和Rx Buffer Length

测试时，RK3399 大核和小核都定为最高频率

表3-1 RK3399 BOX SDK USB MTP传输速率

	Read 16K	Write 16K	Read 64K	Write 64K	Read 1M	Write 1M
USB 2.0	24.7 MBps	22.8 MBps	22.8 MBps	21.5 MBps	24.7 MBps	25.6 MBps
USB 3.0	65.6 MBps	109.0 MBps	71.3 MBps	107.5 MBps	87.3 MBps	110.0 MBps

测试平台：RK3126C样机 Android O + Kernel4.4

表3-2 RK3126C样机USB MTP传输速率

	Read 16K	Write 16K	Read 64K	Write 64K
USB 2.0	14.85 MBps	7.1 MBps	15.15 MBps	7.8 MBps

测试结果分析与性能优化方向：

USB MTP采用的是批量传输的类型，理论上，Rockchip平台USB2.0控制器的批量传输速率约为39MBps（每125微秒传10个transaction，即10 × 512 Bytes），USB3.0控制器的批量传输速率达到400MBps。但MTP的实际传输会受如下几个因素的影响，所以实际的速率会大大低于理论速率。

- CPU的运行频率

  CPU定为高频，MTP的拷贝速率会明显高于CPU变频时的拷贝速率。

- 存储颗粒的读写性能

  如果存储颗粒的读写性能差，会严重影响MTP传输速率。评估存储颗粒的读写性能的方法，请参考2.1 USB Disk传输速率分析

  Kernel4.4 提供了调试接口，打印MTP传输过程中，每次读写存储颗粒(vfs_read/vfs_write)的耗时，使用方法如下：

  查询命令：

  `cat /sys/kernel/debug/usb_mtp/status`

  reset统计状态的命令：

  `echo 0 > /sys/kernel/debug/usb_mtp/status`

- USB控制器的传输性能

  提高USB2控制器的AHB CLK至150MHz，并且DMA Burst设置为最大，可以提高USB控制器的DMA传输性能，从而提高USB控制器的传输速率。

- MTP驱动的Tx/Rx buffer大小

  USB2 控制器（DWC2）支持一次最大传输512KB的数据，USB3控制器（DWC3）支持一次最大传输16MB - 1B，所以，可以尽可能地提高MTP驱动的Tx/Rx buffer大小（当前MTP驱动默认设置为64KB），以减少USB传输的中断数量。建议USB2控制器Tx/Rx buffer设置为64KB，USB3控制器Tx/Rx buffer设置为1MB。

  Kernel4.4的MTP驱动支持通过module parameter配置Tx/Rx buffer，接口如下：

  `/sys/module/usb_f_mtp/parameters/mtp_tx_req_len` （Tx Buffer Length 单位：byte）

  `/sys/module/usb_f_mtp/parameters/mtp_tx_reqs` （Tx Buffer数量）

  `/sys/module/usb_f_mtp/parameters/mtp_rx_req_len` （Rx Buffer Length 单位：byte）

3.2 USB MSC传输速率分析

测试方法：

以Linux-4.4 RK3399 EVB USB3.0 MSC为分析实例

将Mass storage配置为module

```
<*>   USB Gadget Support  --->
      <M>   USB Gadget Drivers
      <M>     Mass Storage Gadget
```

Kernel 增加如下补丁：

```
index f3c7f15..5418d06 100644
--- a/arch/arm64/boot/dts/rockchip/rk3399.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399.dtsi
@@ -404,7 +404,7 @@
                         compatible = "snps,dwc3";
                         reg = <0x0 0xfe800000 0x0 0x100000>;
                         interrupts = <GIC_SPI 105 IRQ_TYPE_LEVEL_HIGH 0>;
-                        dr_mode = "otg";
+                        dr_mode = "peripheral";
                         phys = <&u2phy0_otg>, <&tcphy0_usb3>;
                         phy-names = "usb2-phy", "usb3-phy";
                         phy_type = "utmi_wide";
```

编译内核

编译Mass storage module

`make ARCH=arm64 modules SUBDIRS=drivers/usb/gadget`

生成如下ko

`libcomposite.ko`

`usb_f_mass_storage.ko`

`g_mass_storage.ko`

在系统起来后，先使用dd命令在data目录下创建一个lun0.img，然后加载ko

```
insmod libcomposite.ko
insmod usb_f_mass_storage.ko
insmod g_mass_storage.ko  file=/data/lun0.img removable=1
```

连接PC Ubuntu，格式化lun0，然后在PC端就可以正常访问Mass storage

使用如下脚本测试读速率

```
while true;
do time dd iflag=direct,nonblock if=/media/wlf/4625f014-95f3-419c-96c9-a38f91186eb2/test0 of=/dev/null bs=1024k count=400;
done
```

也可以使用USB3.0 Agilent分析仪测试传输速率，如下图3-1所示

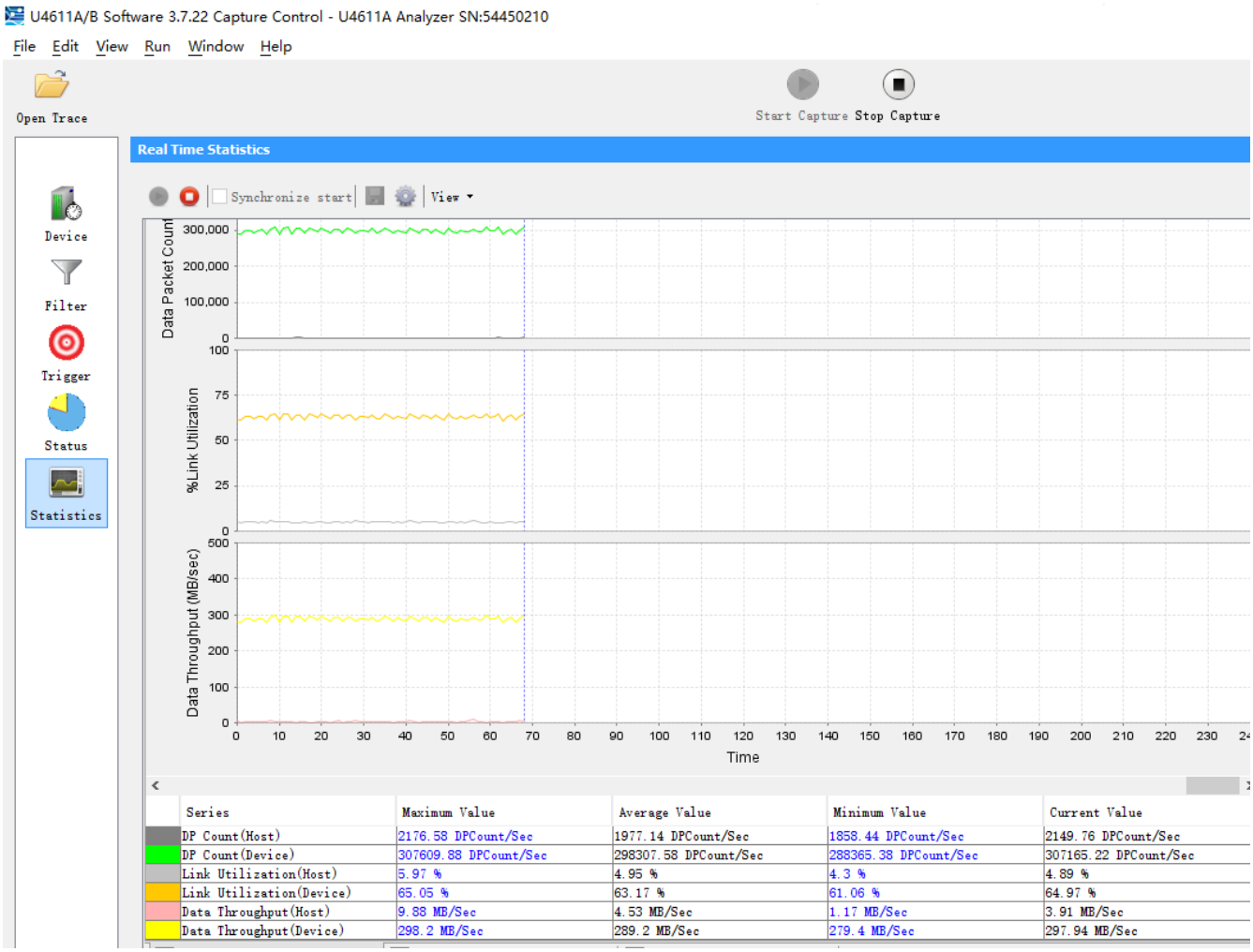

图3-1 USB3 Agilent分析仪测试传输速率

测试结果及分析：

RK3399 变频：读速率 250MBps

RK3399 关闭小核，大核定频1.8：读速率 300MBps

```
3399 关闭小核
echo 0 > /sys/devices/system/cpu/cpu0/online
echo 0 > /sys/devices/system/cpu/cpu1/online
echo 0 > /sys/devices/system/cpu/cpu2/online
echo 0 > /sys/devices/system/cpu/cpu3/online

3399 定频大核
echo userspace > /sys/devices/system/cpu/cpu4/cpufreq/scaling_governor
echo 1800000 >  /sys/devices/system/cpu/cpu4/cpufreq/scaling_setspeed
```

结论：USB MSC的传输速率受CPU运行的频率影响比较明显，CPU频率越高，读速率也越高。

3.3 USB Gadget Webcam传输速率分析

测试方法：

测试工具：

PC Windows： AMCAP软件

PC Ubuntu： guvcview软件 （安装方法：sudo apt install guvcview）

AMCAP和guvcview两款软件都支持实时统计并显示帧率。

测试结果及分析：

如果要支持USB Gadget Webcam，除了内核需要enable UVC gadget驱动，应用层也需要增加UVC的处理。目前，Rockchip平台Kernel-3.10和Kernel-4.4均已经支持usb gadget驱动，应用层只有RV1108 SDK支持UVC gadget，其他芯片平台，可以参考RV1108 SDK或者GitHub的usb gadget uvc driver test application进行修改。

Kernel-3.10 UVC Gadget驱动支持同步传输（Isochronous Transfer）和批量传输（Bulk Transfers）两种传输类型，默认使用同步传输方式，并且默认每个微帧（125微秒）传输1KB。

因此，Kernel-3.10的UVC Gadget可以有4种不同的传输配置方式，不同的配置方式，USB支持的最大带宽和帧率有所不同。

UVC Gadget传输配置方式1：

- 同步传输方式，每个微帧（125微秒）传1KB（默认配置方式）
- USB支持最大带宽：**8000 KB**，即**7.8MB**
- 支持的最大帧率如下表3-3：

表3-3 RV1108 USB Gadget Webcam支持的最大帧率[配置1]

	YUYV格式	**MJPG格式**	**H264格式**
USB 2.0	648×480 @12fps	1920×1080 @30fps	1920×1080 @30fps

UVC Gadget传输配置方式2：

- 同步传输方式，每个微帧（125微秒）传2KB。

  Note：需要更新如下补丁。同时，需要注意，提高传输带宽，可能会引起图像闪屏，如果出现该现象，可以考虑提高USB QOS优先级、提高USB AHB CLK（aclk_peri和hclk_peri）到150MHz以上，以及其他可以提高USB DMA传输效率的方法。

```
diff --git a/drivers/usb/gadget/f_uvc.c b/drivers/usb/gadget/f_uvc.c
index 382536d..17eac12 100644
--- a/drivers/usb/gadget/f_uvc.c
+++ b/drivers/usb/gadget/f_uvc.c
@@ -37,7 +37,7 @@ static unsigned int streaming_interval = 1;
 module_param(streaming_interval, uint, S_IRUGO|S_IWUSR);
 MODULE_PARM_DESC(streaming_interval, "1 - 16");

-static unsigned int streaming_maxpacket = 1024;
+static unsigned int streaming_maxpacket = 2048;
```

- USB支持最大带宽：**16000 KB**，即**15.62MB**
- 支持的最大帧率如下表3-4：

表3-4 RV1108 USB Gadget Webcam支持的最大帧率[配置2]

	YUYV格式	MJPG格式	H264格式
USB 2.0	648×480 @24fps	1920×1080 @30fps	1920×1080 @30fps

UVC Gadget传输配置方式3：

- 同步传输方式，每个微帧（125微秒）传3KB

  Note：需要更新如下补丁。同时，需要注意，提高传输带宽，可能会引起图像闪屏，如果出现该现象，可以考虑提高USB QOS优先级、提高USB AHB CLK（aclk_peri和hclk_peri）到150MHz以上，以及其他可以提高USB DMA传输效率的方法。

```
diff --git a/drivers/usb/gadget/f_uvc.c b/drivers/usb/gadget/f_uvc.c
index 382536d..17eac12 100644
--- a/drivers/usb/gadget/f_uvc.c
+++ b/drivers/usb/gadget/f_uvc.c
@@ -37,7 +37,7 @@ static unsigned int streaming_interval = 1;
 module_param(streaming_interval, uint, S_IRUGO|S_IWUSR);
 MODULE_PARM_DESC(streaming_interval, "1 - 16");

-static unsigned int streaming_maxpacket = 1024;
+static unsigned int streaming_maxpacket = 3072;
```

- USB支持最大带宽：**24000 KB**，即**23.43MB**
- 支持的最大帧率如下表3-5：

表3-5 RV1108 USB Gadget Webcam支持的最大帧率[配置3]

	UYV格式	MJPG格式	H264格式
USB 2.0	648×480 @30fps	1920×1080 @30fps	1920×1080 @30fps

UVC Gadget传输配置方式4：

- 批量传输方式

  Note：需要更新如下补丁。此外，批量传输方式，不支持分辨率的切换，如果要支持多种分辨率，不建议使用批量传输方式。

```
diff --git a/drivers/usb/gadget/f_uvc.c b/drivers/usb/gadget/f_uvc.c
index d146ce7..9eb7aad 100644
--- a/drivers/usb/gadget/f_uvc.c
+++ b/drivers/usb/gadget/f_uvc.c
@@ -45,7 +45,7 @@ static unsigned int streaming_maxburst;
 module_param(streaming_maxburst, uint, S_IRUGO|S_IWUSR);
 MODULE_PARM_DESC(streaming_maxburst, "0 - 15 (ss only)");

-static bool bulk_streaming_ep;
+static bool bulk_streaming_ep = true;
 module_param(bulk_streaming_ep, bool, S_IRUGO | S_IWUSR);
```

- USB支持最大带宽：**35MB**

- 支持的最大帧率如下表3-6：

表3-6 RV1108 USB Gadget Webcam支持的最大帧率[配置4]

	UYV格式	**MJPG格式**	**H264格式**
USB 2.0	648×480 > 30fps	1920×1080 > 30fps	1920×1080 > 30fps

3.4 USB Rndis传输速率分析

USB Rndis实际上就是TCP/IPover USB，就是在USB设备上跑TCP/IP，让USB设备看上去像一块网卡。

测试方法：

测试工具：**iperf**

iperf工具的详细说明，请参考2.2 USB Ethernet传输速率分析

USB Rndis的kernel config设置如下：

- Linux-3.10 Android Gadget USB Rndis配置方法：

```
Device Drivers  --->
   [*] USB support  --->
       <*>   USB Gadget Support  --->
           <*>   USB Gadget Drivers (Android Composite Gadget)  --->
                 Android Composite Gadget
```

对应设备节点：

/sys/class/android_usb/android0/f_rndis 目录下有一些节点可以用来配置网卡的方法和命令

- Linux-3.10 Ethernet Gadget USB Rndis配置方法：

```
Device Drivers  --->
   [*] USB support  --->
       <*>   USB Gadget Support  --->
           <*>   USB Gadget Drivers (Ethernet Gadget (with CDC Ethernet support))  --->
                 Ethernet Gadget (with CDC Ethernet support)
```

- Linux-4.4 Configfs USB Rndis配置方法：

```
Device Drivers  --->
   [*] USB support  --->
       <*>   USB Gadget Support  --->
           <*>   USB Gadget Drivers (USB functions configurable through configfs)  --->
                  [*]       RNDIS
```

实例：

以Linux-3.10 Android Gadget USB Rndis为实例进行分析

USB Rndis的枚举log如下：

```
shell@android:/ # setprop sys.usb.config rndis
[ 2445.809337] rndis_function_bind_config MAC: 36:C8:78:69:C5:D3
[ 2445.809532] android_usb gadget: using random self ethernet address
[ 2445.809666] android_usb gadget: using random host ethernet address
[ 2445.812255] rndis0: MAC 0a:d4:f2:e3:c5:c7
[ 2445.812330] rndis0: HOST MAC ea:88:5c:93:9d:b5    如连接window，则对应windows以太网卡MAC
[ 2446.208233] DWC_OTG: ********soft
connect!!!**************************************
[ 2446.541253] android_usb gadget: high speed config #1: android
[ 2446.542681] android_work: sent uevent USB_STATE=CONFIGURED
```

设置IP并启动rndis：

`ifconfig rndis0 192.168.1.2 up`

其中， IP地址192.168.1.2 要与PC的IP 在同一个网段， 也即都在192.168.XX.XX网段。

更详细的配置和测试方法，请参考2.2 USB Ethernet传输速率分析

Note：测试前，需要关闭Windows防火墙，否则可能导致Rockchip平台无法pingWindows。

测试结果：

以下均为High-speed的测试结果，Super-Speed暂时没有测试过。

Linux-3.10 Android Gadget USB Rndis：上行**20Mbps**，下行**40Mbps**

Linux-3.10 Ethernet Gadget USB Rndis：上行和下行均为**100Mbps**

Linux-4.4 Configfs USB Rndis：上行和下行均为**100Mbps**

测试结果分析及优化方向：

目前，已知的一个问题是Linux-3.10 Android Gadget USB Rndis上行和下行速率都偏低，需要进一步优化。

3.5 USB Gadget HID传输速率分析

测试方法：

测试工具：

PC Windows：HidTest.exe，PortHelper1.7.exe，或者下载HID API

PC Ubuntu：Linux-3.10 Documentation/usb/gadget_hid.txt中的hid_gadget_test测试脚本

或者下载HID API

测试结果：

测试rk3188 Gadget HID传输速率，结果如下：

case1. 设置report_length为512 bytes，传输间隔为1ms，传输文件大小为10768KB，HID为high speed设备

PC -> 3188 速率： 249 KBps

3188 -> PC 速率：125 KBps

case2. 设置report_length为512 bytes，传输间隔为125us，传输文件大小为10768KB，HID为high speed设备

PC -> 3188 速率： 872 KBps 3188 -> PC 速率： 2659 KBps

测试结果分析：

USB Gadget HID的传输速率主要由HID描述符的**report_length**的长度和**中断传输间隔**决定。Linux USB Gadget HID驱动源文件（drivers/usb/gadget/hid.c）中，没有提供HID描述符，需要开发者参考Linux文档Documentation/usb/gadget_hid.txt进行开发。如下给出HID report_length设置为512bytes的HID描述符参考代码。

```
index 79afa82..a05ad79 100644
--- a/drivers/usb/gadget/hid.c
+++ b/drivers/usb/gadget/hid.c
@@ -271,10 +272,93 @@ MODULE_DESCRIPTION(DRIVER_DESC);
 MODULE_AUTHOR("Fabien Chouteau, Peter Korsgaard");
 MODULE_LICENSE("GPL");

+static struct hidg_func_descriptor rk_hidg_desc = {
+   .subclass       = 0, /* No subclass */
+   .protocol       = 2, /* Keyboard */
+   .report_length      = 512,
+   .report_desc_length = 38,
+   .report_desc        = {
+       0x06, 0x00, 0xFF,    /*USAGE_PAGE (Vendor Defined Page 1)*/
+       0x09, 0x01,     /*USAGE (Vendor Usage 1)*/
+       0xA1, 0x01,         /*COLLECTION (Application)*/
+       0x09, 0x02,     /*  USAGE (Vendor Usage 2)*/
+       0x09, 0x03,     /*USAGE (Vendor Usage 3)*/
+       0x15, 0x00,         /*LOGICAL_MINIMUM (0)*/
+       0x26, 0xFF, 0x00, /*LOGICAL_MAXIMUM (255)*/
+       0x75, 0x10,         /*REPORT_SIZE (16)*/
+       0x96, 0x00, 0x01, /* REPORT_COUNT (256) */
+       0x81, 0x02,         /*INPUT (Data,Var,Abs)  */
+       0x09, 0x04,     /*USAGE (Vendor Usage 4)*/
+       0x15, 0x00,         /*LOGICAL_MINIMUM (0)   */
+       0x26, 0xFF, 0x00, /*LOGICAL_MAXIMUM (255)   */
+       0x75, 0x10,         /*REPORT_SIZE (16)*/
+       0x96, 0x00, 0x01,   /*REPORT_COUNT (256)*/
+       0x91, 0x02,         /*OUTPUT (Data,Var,Abs) */
+       0xC0            /*END_COLLECTION*/
+   }
+};
```

如果要支持report_length 为1024bytes，可以改为

```
@@ -275,7 +275,7 @@ MODULE_LICENSE("GPL");
 static struct hidg_func_descriptor rk_hidg_desc = {
    .subclass       = 0, /* No subclass */
    .protocol       = 2, /* Keyboard */
-   .report_length      = 512,
+   .report_length      = 1024,
    .report_desc_length = 38,
```

```
    .report_desc        = {
        0x06, 0x00, 0xFF,    /*USAGE_PAGE (Vendor Defined Page 1)*/
@@ -285,13 +285,13 @@ static struct hidg_func_descriptor rk_hidg_desc = {
        0x09, 0x03,     /*USAGE (Vendor Usage 3)*/
        0x15, 0x00,         /*LOGICAL_MINIMUM (0)*/
        0x26, 0xFF, 0x00, /*LOGICAL_MAXIMUM (255)*/
-       0x75, 0x10,         /*REPORT_SIZE (16)*/
+       0x75, 0x20,         /*REPORT_SIZE (32)*/
        0x96, 0x00, 0x01, /* REPORT_COUNT (256) */
        0x81, 0x02,         /*INPUT (Data,Var,Abs)  */
        0x09, 0x04,     /*USAGE (Vendor Usage 4)*/
        0x15, 0x00,         /*LOGICAL_MINIMUM (0)   */
        0x26, 0xFF, 0x00, /*LOGICAL_MAXIMUM (255)   */
-       0x75, 0x10,         /*REPORT_SIZE (16)*/
+       0x75, 0x20,         /*REPORT_SIZE (32)*/
        0x96, 0x00, 0x01,   /*REPORT_COUNT (256)*/
        0x91, 0x02,         /*OUTPUT (Data,Var,Abs) */
        0xC0            /*END_COLLECTION*/
```

而HID的中断传输间隔，可以修改drivers/usb/gadget/f_hid.c文件里面的hidg_hs_in_ep_desc和hidg_hs_out_ep_desc结构体的成员bInterval，原生是4（1ms），最小可以改为1（125us）。