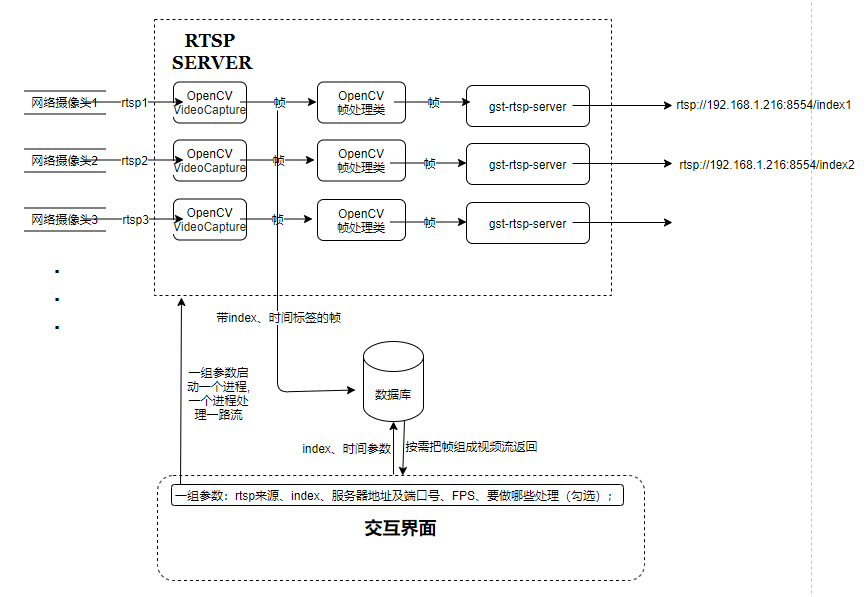
# RTSP流媒体服务器

## 简介

本实验拟通过利用opencv + gstreamer + gst-rtsp-server等搭建RTSP流媒体服务器实现对来自网络摄像头的RTSP视频流进行管理；

系统数据流图如下所示：



系统数据流图

通过参数设置，在需要帧数据的时候从互斥线程锁中获取拉流线程当前帧；同时在组流+推流线程中做图像处理，对组流需要的帧作处理。每一路rtsp流发送到唯一的URL地址；其中，每一路视频流的管道及参数输入如下图所示：

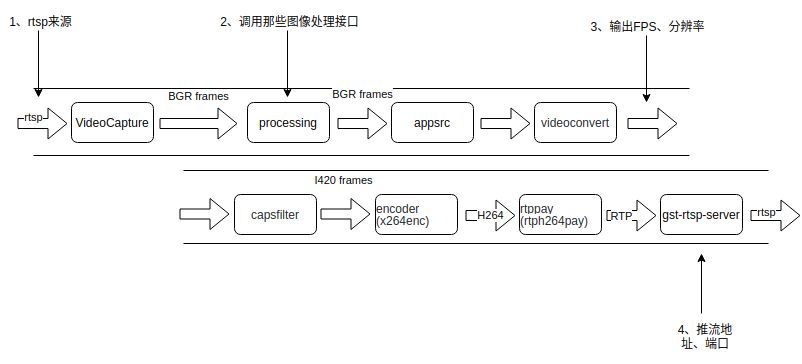


图 系统pipeline

每一路流都需要通过一组参数来配置，通过定义一个结构体来存储所有输入参数：

typedef struct

{

string INDEX; //rtsp标号

string in\_rtsp; //来源uri

int out\_width; //组流分辨率

int out\_height;

int out\_fps; //组流帧率

string out\_port; //推流端口

} Params;

每一个管道（即每一路rtsp流）需要启动一个进程来管理，一个进程包括两个线程；两个线程之间通过一个全局的静态的线程互斥锁来传递帧数据。

1.拉流线程：利用opencv videocapture 源源不断地接受每一帧数据；

2.处理+组流+推流线程；

## 实验目的

1. 利用opencv实现RTSP解码；
2. 实现对解码出的Mat帧进行处理，如区域选择；
3. 实现对解码出的Mat帧进行本地存储；
4. 基于Gstreamer实现对Mat帧进行H.264编码、组流至指定服务器url；
5. 利用互斥线程锁将解码、处理、编码、组流等步骤封装成管线，每一路管线对应一路RTSP流；
6. 利用多进程管理实现同时处理多路流；
7. 利用CUDA编程加速RTSP解码，解放CPU资源、降低延迟；
8. 通过init配置文件启动系统，其中包括rtsp路数以及每一路流各自的参数；

## 实验环境

操作系统Ubuntu18.04

显卡：1080Ti

显卡驱动：NVIDIA-DRIVER-450.80.02

编程语言：C/C++

CUDA10.1、cuDNN7.6.3、nvidia video codec sdk 9.1.23、nv-codec-headers9.1、Opencv4.2、Gstreamer1.0、gst-rtsp-server等

## 实验方法

### 环境配置、编译

#### 安装显卡驱动

sudo apt-get install nvidia-drivers-450

通过nvidia-msi可以验证是否安装成功。

#### 安装CUDA10.1

#下载

Wget <http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_installers/cuda_10.1.243_418.87.00_linux.run>

#安装

sudo bash cuda\_10.1.243\_418.87.00\_linux.run

# 安装的时候取消driver选项！

# 配置环境变量

sudo gedit ~/.bashrc

export PATH="/usr/local/cuda-10.1/bin:$PATH"

export LD\_LIBRARY\_PATH="/usr/lcoal/cuda-10.1/lib64:$LD\_LIBRARY\_PATH"

source ~/.bashrc

# 验证是否安装成功

Nvcc -V

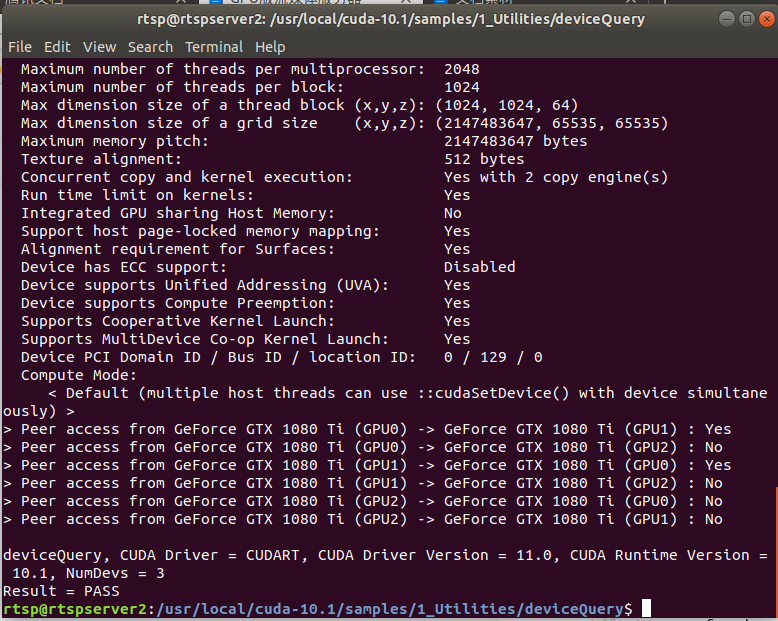
或

cd /usr/local/cuda-10.1/samples/1\_Utilities/deviceQuery

sudo make

./deviceQuery

# 安装成功如下图所示



#### 安装cuDNN7.6.3

# https://developer.nvidia.com/rdp/cudnn-archive

#下载cudnn后解压

tar -xzvf cudnn-10.1-linux-x64-v7.6.3.30.tgz

# 拷贝.h 和 libs文件到cuda安装目录，并给予执行权限：

sudo cp cuda/include/cudnn.h /usr/local/cuda/include

sudo cp cuda/lib64/libcudnn\* /usr/local/cuda/lib64

sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn\*

# 如下形式创建软连接

sudo ln -sf /usr/local/cuda-10.1/targets/x86\_64-linux/lib/libcudnn.so.7.6.3 /usr/local/cuda-10.1/targets/x86\_64-linux/lib/libcudnn.so.7

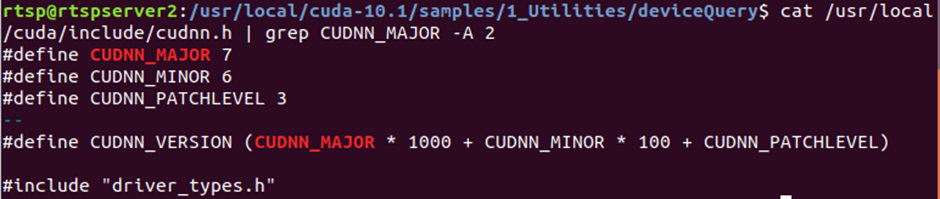
sudo dpkg -i libcudnn7\_7.6.3.30-1+cuda10.1\_amd64.deb

sudo dpkg -i libcudnn7-dev\_7.6.3.30-1+cuda10.1\_amd64.deb

sudo dpkg -i libcudnn7-doc\_7.6.3.30-1+cuda10.1\_amd64.deb

#测试

cat /usr/local/cuda/include/cudnn.h | grep CUDNN\_MAJOR -A 2



#### nvidia video codec sdk 9.1.23

#https://developer.nvidia.com/video-codec-sdk-archive

下载nvidia video codec sdk 9.1.23

解压缩之后把  Video\_Codec\_SDK\_9.1.23/include/ 下面的

cuviddec.h 和 nvcuvid.h文件拷贝到/usr/include下面就好了

#### ffmpeg以及nv-codec-headers9.1

#依赖

sudo apt-get update

sudo apt-get -y install autoconf automake build-essential libass-dev libfreetype6-dev libsdl2-dev libtheora-dev libtool libva-dev libvdpau-dev libvorbis-dev libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev pkg-config texinfo zlib1g-dev

#安装yasm  汇编编译器，编译某些依赖库的时候需要

sudo apt-get install yasm -y

# 安装lib264 H.264视频编码器，如果需要输出H.264编码的视频就需要此库，所以可以说是必备

sudo apt-get install libx264-dev -y

#H.265/HEVC视频编码器。如果不需要此编码器，可以跳过，并在ffmpeg的configure命令中移除--enable-libx265

sudo apt-get install libx265-dev -y

#安装 libvpx VP8/VP9视频编/解码器 如果不需要此编/解码器，可以跳过，并在ffmpeg的configure命令中移除--enable-libvpx

sudo apt-get install libvpx-dev -y

#安装 安装libfdk-aac AAC音频编码器，必备

sudo apt-get install libfdk-aac-dev -y

#安装libmp3lam MP3音频编码器，必备

sudo apt-get install libmp3lame-dev -y

#OPUS音频编码器,如果不需要此编码器，可以跳过，并在ffmpeg的configure命令中移除--enable-libopus

sudo apt-get install libopus-dev -y

**#安装NVENC：需要先更新源,再依次安装**

https://blog.csdn.net/weixin\_44354586/article/details/89395892

sudo apt-get -y install glew-utils

sudo apt-get -y install libglew-dbg

sudo apt-get -y install libglew-dev

sudo apt-get -y install libglew2.0

sudo apt-get -y install libglewmx-dev

sudo apt-get -y install libglewmx-dbg

sudo apt-get -y install freeglut3=2.8.1-1

sudo apt-get -y install freeglut3-dev=2.8.1-1

sudo apt-get -y install freeglut3-dbg=2.8.1-1

sudo apt-get -y install libghc-glut-dev

sudo apt-get -y install libghc-glut-doc

sudo apt-get -y install libghc-glut-prof

sudo apt-get -y install libalut-dev

sudo apt-get -y install libxmu-dev

sudo apt-get -y install libxmu-headers

sudo apt-get -y install libxmu6

sudo apt-get -y install libxmu6-dbg

sudo apt-get -y install libxmuu-dev

sudo apt-get -y install libxmuu1

sudo apt-get -y install libxmuu1-dbg

git clone https://github.com/FFmpeg/FFmpeg ffmpeg -b master

#9.1： Linux nvidia driver 435.21 or newer

git clone https://github.com/FFmpeg/nv-codec-headers nv-codec-header -b sdk/9.1

cd nv-codec-headers

make

sudo make install

#编译ffmpeg

cd ffmpeg

./configure --prefix=/usr/local/ffmpeg --disable-asm --disable-x86asm \ --enable-cuda --enable-cuvid --enable-nvenc \ --enable-nonfree --enable-libnpp \ --extra-cflags=-I/usr/local/cuda/include \ --extra-cflags=-fPIC --extra-ldflags=-L/usr/local/cuda/lib64 \ --enable-gpl --enable-libx264 --enable-libx265 \ --enable-shared \ --enable-libass \ --enable-libfdk-aac \ --enable-libfreetype \ --enable-libmp3lame \ --enable-libopus \ --enable-libtheora \ --enable-libvorbis

make -j8

sudo make -j8 install

make -j8 distclean

hash -r #清除缓存

sudo gedit /etc/ld.so.conf.d/ffmpeg.conf 添加: /usr/local/ffmpeg/lib

sudo ldconfig

sudo gedit ~/.bashrc

#加入以下内容

export PATH=/usr/local/ffmpeg/bin:$PATH

export FFMPEG\_HOME=/usr/local/ffmpeg

export PATH=$FFMPEG\_HOME/bin:$PATH

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/lib

export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/ffmpeg/lib/pkgconfig:/usr/local/lib

source ~/.bashrc

#### opencv4.2.0+opencv\_contrib-4.2.0

#依赖

sudo apt-get update -y

sudo apt-get remove -y x264 libx264-dev # Remove the older version of libx264-dev and x264

sudo apt-get install -y build-essential checkinstall cmake pkg-config yasm

sudo apt-get install -y git gfortran

sudo add-apt-repository -y "deb http://security.ubuntu.com/ubuntu xenial-security main"

sudo apt-get install -y libjpeg8-dev libjasper-dev libpng12-dev

sudo apt-get install -y libtiff5-dev

sudo apt-get install -y libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev

sudo apt-get install -y libxine2-dev libv4l-dev

sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev

sudo apt-get install -y qt5-default libgtk2.0-dev libtbb-dev

sudo apt-get install -y libatlas-base-dev

sudo apt-get install -y libfaac-dev libmp3lame-dev libtheora-dev

sudo apt-get install -y libvorbis-dev libxvidcore-dev

sudo apt-get install -y libopencore-amrnb-dev libopencore-amrwb-dev

sudo apt-get install -y x264 v4l-utils

# Some Optional Dependencies

sudo apt-get install -y libprotobuf-dev protobuf-compiler

sudo apt-get install -y libgoogle-glog-dev libgflags-dev

sudo apt-get install -y libgphoto2-dev libeigen3-dev libhdf5-dev doxygen

sudo apt-get install libgtkglext1=1.2.0-3.1fakesync3 -y

libgtkglext1-dev=1.2.0-3.1fakesync3

libgtkglext1-dbg=1.2.0-3.1fakesync3

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

sudo apt-get install checkinstall yasm libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev libtbb-dev libqt4-dev libgtk2.0-dev libmp3lame-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils

sudo apt-get install libcanberra-gtk-module

# 编译及测试按照以下文档

cd opencv

mkdir build

cd build

sudo cmake -D CMAKE\_INSTALL\_PREFIX=/usr/local/opencv4 \

-D CMAKE\_BUILD\_TYPE=Debug \

-D OPENCV\_EXTRA\_MODULES\_PATH=../opencv\_contrib4/modules \

-D BUILD\_opencv\_hdf=OFF \

-D BUILD\_opencv\_python3=ON \

-D WITH\_CUDA=ON \

-D WITH\_OPENGL=ON \

-D WITH\_OPENMP=ON \

-D WITH\_GTK=ON \

-D WITH\_OPENCL=ON \

-D WITH\_VTK=ON -D WITH\_TBB=ON \

-D WITH\_GSTREAMER=ON \

-D WITH\_CUDNN=ON \

-D WITH\_CUBLAS=ON \

-D WITH\_GTK\_2\_X=ON \

-D BUILD\_EXAMPLES=ON \

-D OPENCV\_ENABLE\_NONFREE=ON \

-D WITH\_FFMPEG=ON \

-D OPENCV\_GENERATE\_PKGCONFIG=ON \

-D WITH\_NVCUVID=ON \

-D CUDA\_TOOLKIT\_ROOT\_DIR=/usr/local/cuda-10.2 \

-D CUDA\_ARCH\_BIN=6.1

make -j12

sudo make install -j12

sudo gedit /etc/ld.so.conf.d/opencv.conf

/usr/local/opencv4/lib

sudo ldconfig

sudo gedit /etc/bash.bashrc

//在末尾添加

PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/opencv4/lib/pkgconfig #根据opencv的lib设置的路径更改export PKG\_CONFIG\_PATH

source /etc/bash.bashrc//激活配置然后更新database

sudo updatedb

gedit ~/.bashrc

export PKG\_CONFIG\_PATH=/usr/local/opencv-4.2.0/lib/pkgconfig

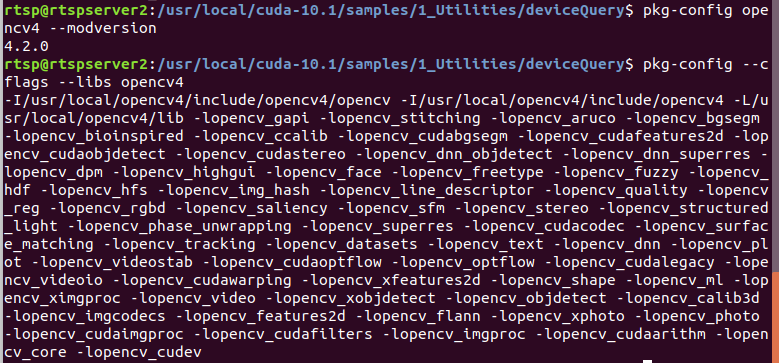
export LD\_LIBRARY\_PATH=/usr/local/opencv-4.2.0/lib

source ~/.bashrc

# 分别运行如下两条命令，验证opencv是否交叉编译成功

pkg-config opencv4 --modversion

pkg-config --cflags --libs opencv4



#### 安装gstreamer

Gstreamer是一个用于开发流媒体应用的开源框架，采用了基于插件（plugin）和管道（pipeline）的体系结构，框架中的所有的功能模块都被实现成可以插拔的组件（component）， 并且能够很方便地安装到任意一个管道上。由于所有插件都通过管道机制进行统一的数据交换，因此很容易利用已有的各种插件“组装”出一个功能完善的流媒体应用程序。deepstream是基于gstreamer的。

# 操作系统需要>=ubuntu18.04，否者有些依赖不能完整安装

sudo apt-get install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio

sudo apt --fix-broken install

#### 安装gst-rtsp-server

1.下载gst-rtsp-server源码

git clone https://github.com/GStreamer/gst-rtsp-server.git

2.下载GStreamer/common源码

cd gst-rtsp-server

git clone <https://github.com/GStreamer/common.git>

3.编译

先安装依赖

sudo apt-get install gtk-doc-tools

sudo apt-get install libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-dev libgstreamer-plugins-bad1.0-dev

git checkout remotes/origin/1.8

./autogen.sh

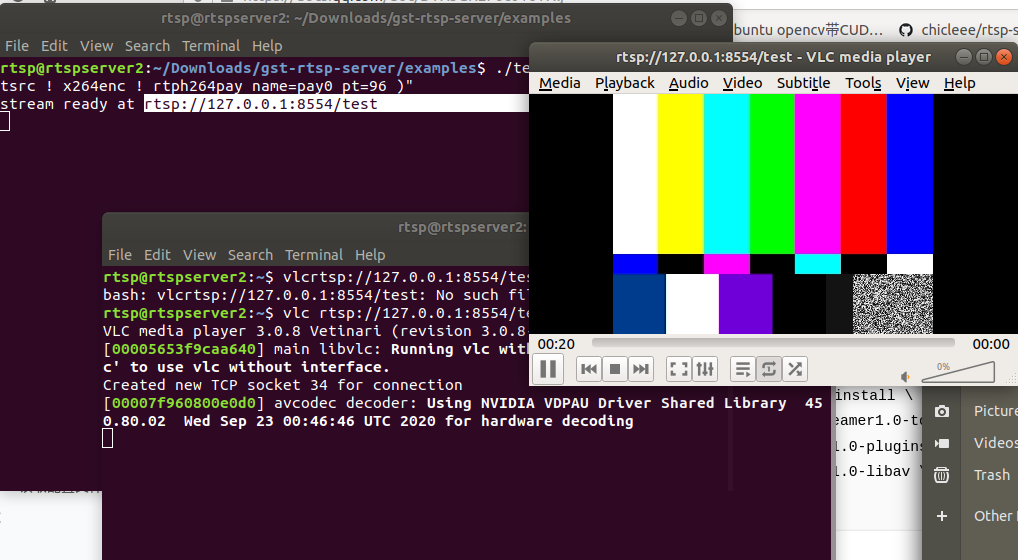
make

make install

如果报错，安装依赖项：sudo apt install \ libssl1.0.0 \ libgstreamer1.0-0 \ gstreamer1.0-tools \ gstreamer1.0-plugins-good \ gstreamer1.0-plugins-bad \ gstreamer1.0-plugins-ugly \ gstreamer1.0-libav \ libgstrtspserver-1.0-0

4测试examples下例子：

./test-launch "( videotestsrc ! x264enc ! rtph264pay name=pay0 pt=96 )"



#### 安装boost管理配置文件

sudo apt-get install libboost-all-dev

### 实验程序

代码已上传至：<https://github.com/chicleee/rtsp-server>

git clone <https://github.com/chicleee/rtsp-server>

cd rtsp-server

cd src

其中有以下重要文件，

* Test\_gpu文件夹是测试交叉编译环境是否配置正确的代码，直接

make

./main.out rtsp://

* Start.cpp是只使用CPU的版本

编译命令：g++ -std=c++11 start.cpp -o start -lboost\_system -lboost\_filesystem -lpthread -lgstapp-1.0 `pkg-config --libs --cflags opencv gstreamer-1.0 gstreamer-rtsp-server-1.0`

运行命令：./start

* start\_gpu.cpp是GPU版本

编译命令：g++ -std=c++11 start\_gpu.cpp -g -o start\_gpu -lboost\_system -lboost\_filesystem -lpthread -lgstapp-1.0 `pkg-config --libs --cflags opencv4 gstreamer-1.0 gstreamer-rtsp-server-1.0` \

-I/usr/local/opencv4/include/opencv4/opencv2 \

-I/usr/local/cuda/include \

-L/usr/local/cuda/lib64 \

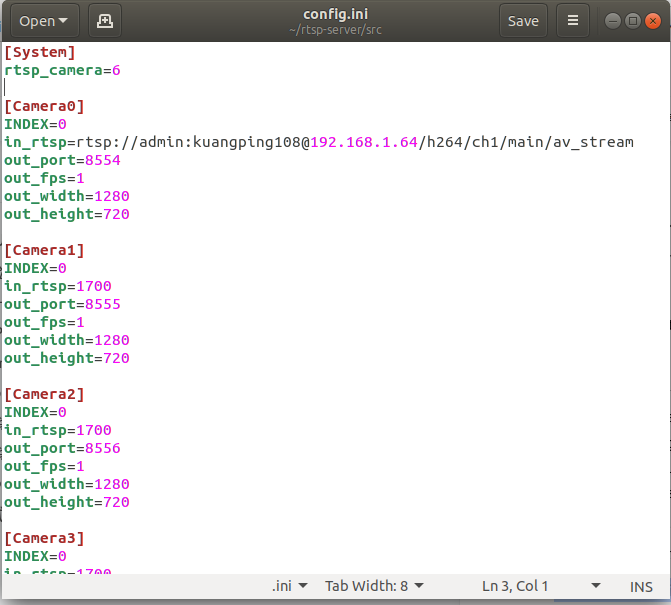
-I/usr/include/eigen3 \

-L/usr/lib/x86\_64-linux-gnu -lcuda -ldl -lnvcuvid

运行命令：

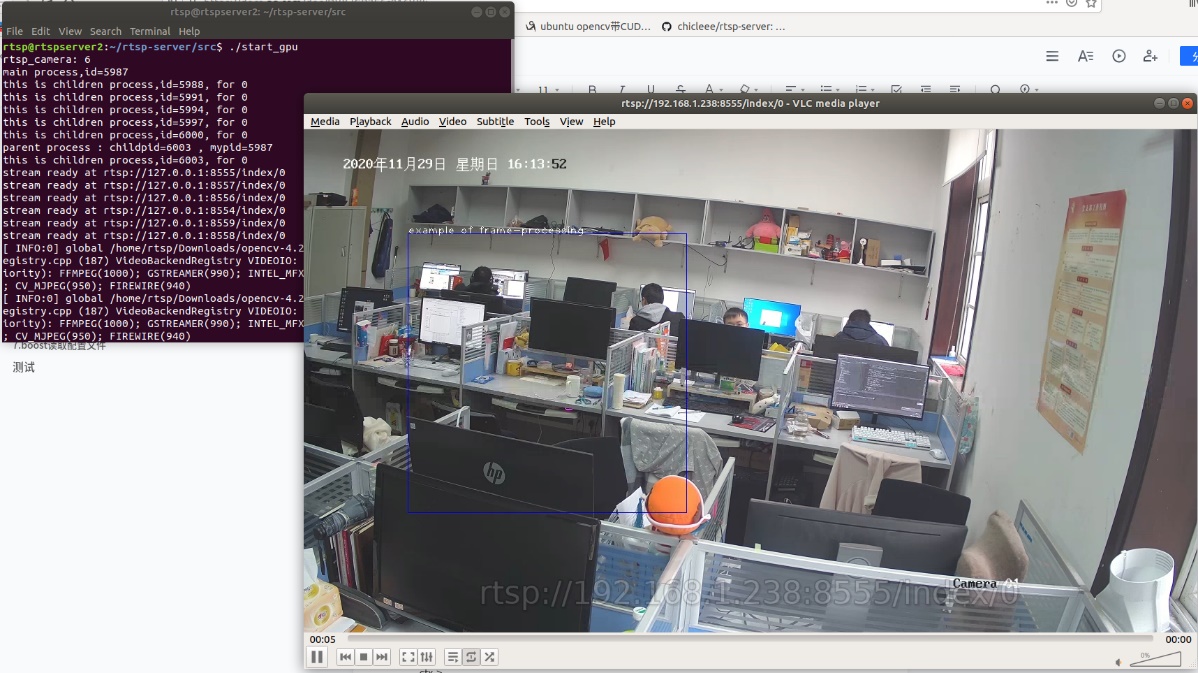
./start\_gpu

* Config.ini是系统的配置文件，包含输入参数如下图所示：

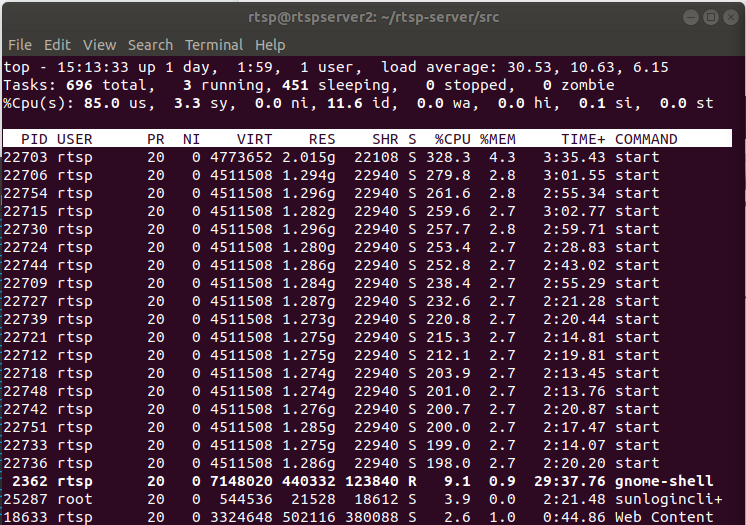


## 实验结果

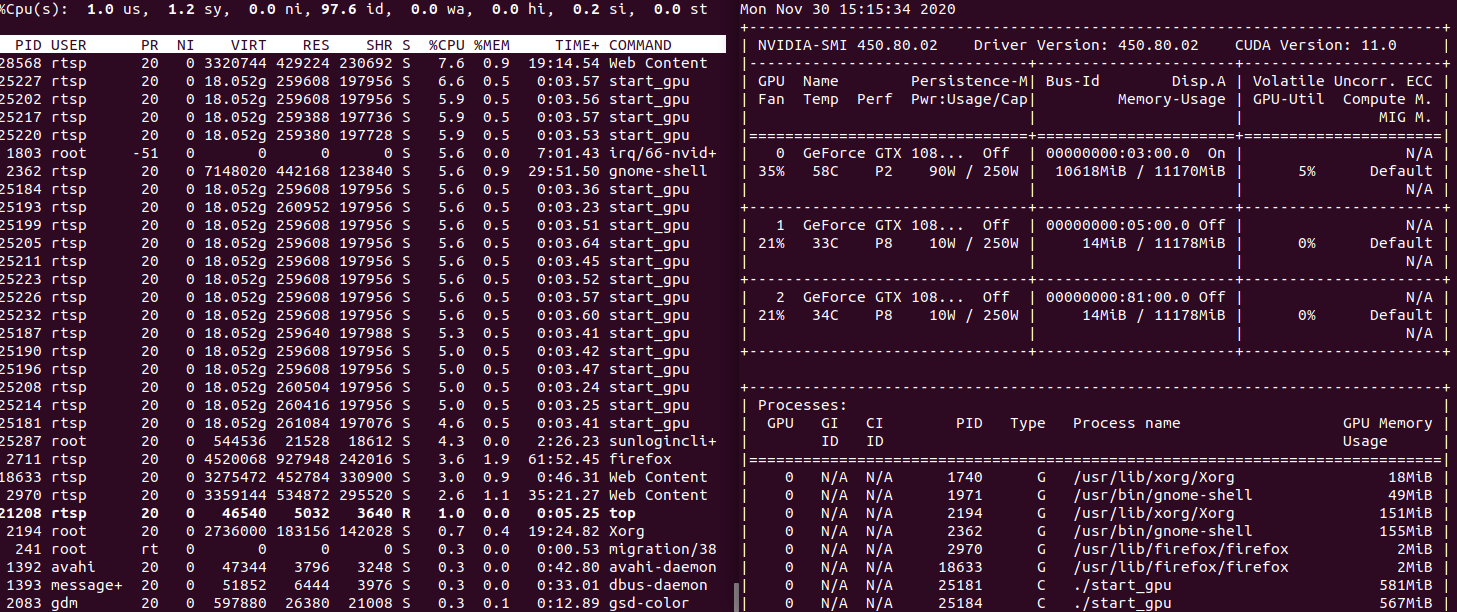
测试运行结果如下，在服务器端通过./start\_gpu启动程序，客户端在<http://192.168.1.238:8554/index/0>可以获取我们推的流：



下图是./start启动CPU版本的RTSP流媒体服务器同时处理18路流是CPU的负载情况，CPU严重超载：



下图是运行./start\_gpu，用GPU解码的RTSP流媒体服务器处理18路流时CPU的负载情况以及GPU显存消耗情况：



可以看到，在使用GPU解码的系统极大地解放了CPU资源，但同时在声明nvidia提供的流接口时就占用来580M显存（每路4K RTSP流）

## 结论

本RTSP流媒体服务器实现了预期的系统功能，但由于GPU解码一路4K的RTSP视频流需要占用580M的显存，解码一路1080P的视频流需要约200M的显存，故在11GB显存的1080Ti显卡上，最多只能同时处理18路4K流、或者50路1080P流。由于目前的应用需求对路数要求高，暂搁置。