



技术开启新“视”界
Technology Bring New Vision

解密GPU：视频转码与分析加速

季光

NVIDIA 高级工程师 gji@nvidia.com

2018.10.20

关于我

- 2014年加入NVIDIA，现为高级工程师
- 云游戏基础设施开发项目负责人
- Video Codec SDK v8.1应用层主要开发者
- DeepStream SDK v1.0主要开发者



本次分享的目标

作为一份来自N粉的“GPU视频处理技术调研报告”

- 针对技术选型中的用户：为GPU的采纳与部署提供技术依据与路线
- 针对既有的GPU用户：提供前沿、全面的进阶信息，丰富知识储备

NVIDIA GPU产品线

- 利用少量的GPU型号，衍生出复杂的显卡型号
 - 经芯片筛选(SM数量与频率)，搭配不同容量的显存和不同的驱动，得到不同产品
- 以TU104为例
 - 消费级： GeForce RTX 2080 (8 GB)
 - 企业级： Tesla T4 (16 GB)
 - 工作站： Quadro RTX 5000 (16 GB)
- 其他Turing GPU型号
 - TU102: GeForce RTX 2080 Ti, Quadro RTX 6000/8000
 - TU106: GeForce RTX 2070

寻找最实惠的显卡型号

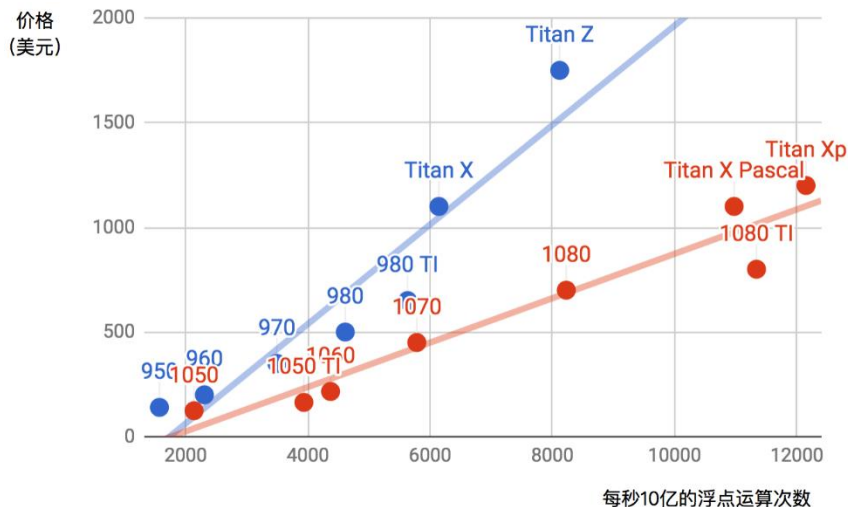
我喜欢精打细算，我就想要多快好省，怎么选显卡？

考虑因素

- 单位计算力的价格 (RMB/FLOPS)
- 但显存小、CUDA核数少会制约使用场景

经验

- 买新不买旧
- 关注消费级产品的高端型号





姑且选GeForce RTX 2080

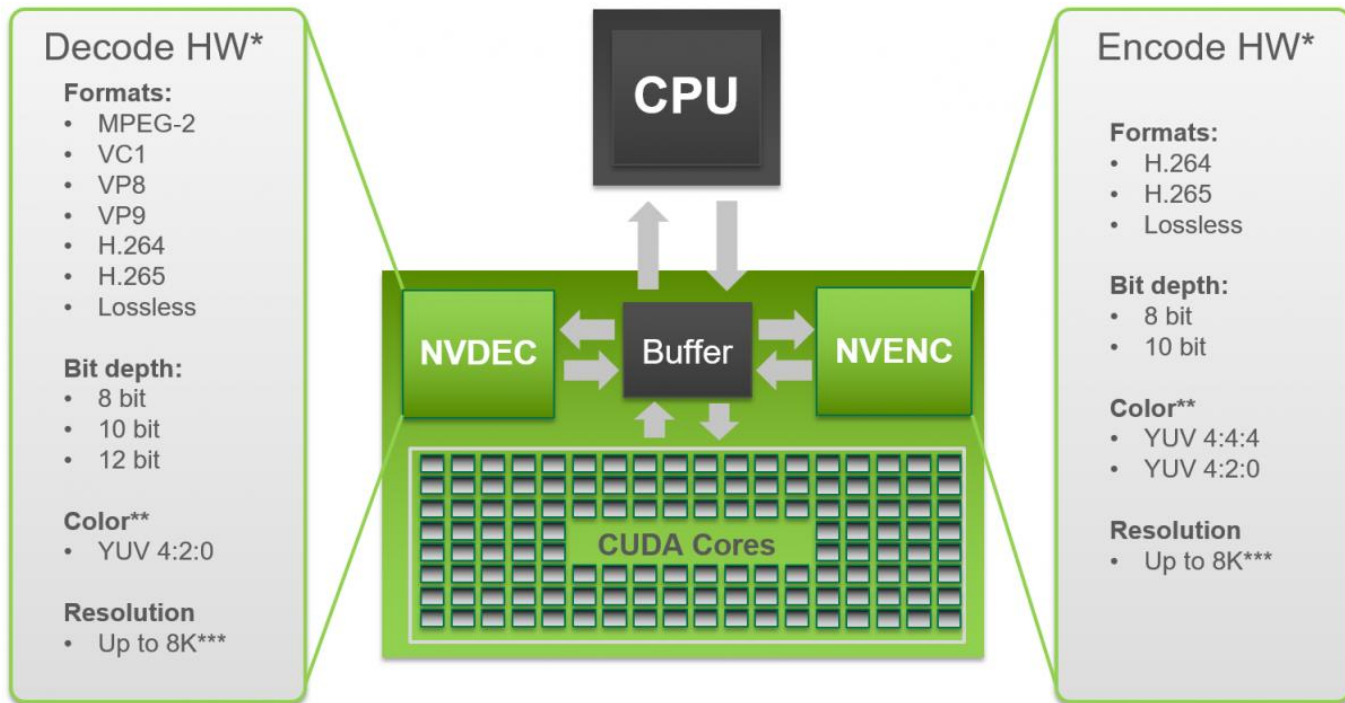
- RMB 6299 (2018.10.20)
- 次旗舰（上有RTX 2080 Ti）
- GPU
 - TU104
 - 46 SM (2944 CUDA Cores)
 - 1.515 (1.71) GHz
 - 8920 GFLOPS
- 8 GB GDDR6显存

对比上一代性价比之王1080 Ti

- RMB 5499
- 10609 GFLOPS / 11 GB GDDR5X显存



NVIDIA GPU的视频处理模块



NVIDIA GPU API



Easy access to GPU
video acceleration

DeepStream SDK

cuDNN, TensorRT,
cuBLAS, cuSPARSE

VIDEO CODEC SDK

Video Encode and Decode for Windows and Linux
CUDA, DirectX, OpenGL interoperability

CUDA TOOLKIT

APIs, libraries, tools, samples

NVIDIA DRIVER

NVENC

Video encode



NVDEC

Video decode

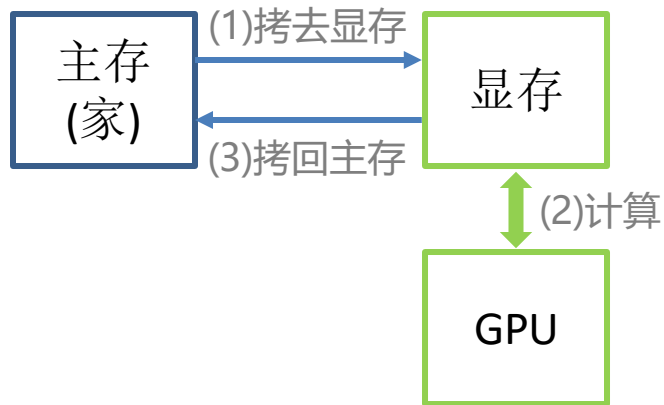


CUDA

High-performance
computing on GPU



GPU编程基础



```
__global__ void square(float *p_a, float *p_b) {
    *p_b = *p_a * *p_a;
}

int main(void) {
    //在GPU上计算 b = a**2

    //主存上的a, b
    float a = 2.0f, b;

    //显存上的a, b
    float *dp_a = NULL, *dp_b = NULL;
    cudaMalloc(&dp_a, sizeof(a));
    cudaMalloc(&dp_b, sizeof(b));

    //(1)将a拷去显存
    cudaMemcpy(dp_a, &a, sizeof(a), cudaMemcpyHostToDevice);
    //(2)计算
    square<<<1,1>>>(dp_a, dp_b);
    //(3)将b拷回主存
    cudaMemcpy(&b, dp_b, sizeof(b), cudaMemcpyDeviceToHost);


    //释放分配的显存
    cudaFree(dp_a);
    cudaFree(dp_b);

    printf("%f\n", b);
    return 0;
}
```

GPU编程基础

- CUDA SDK提供了基本的GPU编程API
- 显存管理
 - 分配、释放显存: `cudaMalloc/cudaFree`
- 主存-显存数据交换
 - 数据在主存与显存之间的拷贝: `cudaMemcpy`, 参数指明两个地址以及数据拷贝方向
- CUDA C编程: 自己写CUDA kernel
- 预置API
 - GPU上的图像处理以及计算, 多可用NV官方预先实现的API完成
 - 常用API: `NPP/cuDNN/cuBLAS/TensorRT`

图像处理加速

- 基本原理：将图像载入显存，利用GPU的并行计算能力进行处理
 - NPP (NVIDIA Performance Primitives)
 - 实现了图像处理常用算法
 - 通过API直接使用，无需GPU编程（但需理解主存-显存数据交换）
 - CUDA C编程
 - NVIDIA Codec SDK提供了缩放、颜色空间转换等常用算法的样例代码
 - 入门容易
 - 性能优化套路多
- 



视频解码

- 支持硬件
 - 全产品线，并发路数无限制
- 支持格式
 - Tesla P4: H264 (4Kx4K); HEVC 8/10/12-bit (8Kx8K); VP9; **VP8**; MPEG4; MPEG2; VC1
 - Tesla P40: H264 (4Kx4K); HEVC 8/10/12-bit (8Kx8K); VP9 **8/10/12-bit**; MPEG4; MPEG2; VC1
 - Tesla T4/RTX 2080: 所有Tesla P4 & P40支持的格式
- GPU解码速度与格式、分辨率、码率以及B帧数量有关
 - 以Tesla P4为例: 1.5 Mbps 1080p 无B帧H264/HEVC解码速度分别为678/733 FPS
 - RTX 2080: 与Tesla P4接近
 - Tesla T4: 1391/2376 FPS





视频解码实战

- FFmpeg命令行

- 编译时加选项，启用GPU解码、编码与中间处理
- enable-nonfree --enable-cuda --enable-nvenc
--enable-cuvid --enable-libnpp
--extra-cflags=-I/usr/local/cuda/include
--extra-ldflags=-L/usr/local/cuda/lib64
- ffmpeg -c:v h264_cuvid -i in.h264 -f null -

- API

- NVDecode (原名NVCUVID)
- NvDecLite
- FFmpeg API

```
NvDecLite dec(cuContext, false, cudaVideoCodec_H264);
while (true) {
    int nRead = fread(pBuf, 1, nBuf, fpH264);
    dec.CacheAppend(pBuf, nRead);
    uint8_t *apFrame[8];
    int nFrameLimit = sizeof(apFrame)/sizeof(apFrame[0]);
    int nFrameReturned = 0;
    while (dec.CacheDecode(apFrame, nFrameLimit,
        &nFrameReturned, nRead == 0)) {
        for (int i = 0; i < nFrameReturned; i++) {
            fwrite(apFrame[i], 1, dec.GetFrameSize, fpNv12);
        }
    }
    if (!nRead) {
        break;
    }
}
```



视频编码

- 支持硬件
 - Tesla/Quadro无并发路数限制
 - GeForce整个主机限2路
- 画质
 - 上一代产品Pascal接近x264 medium预设
 - 本代产品在保持画质相当的情况下，节省码流10~20%
 - HEVC编码支持B帧
- 编码速度：与参数设置有关
 - 以Tesla P4为例：1080p H264 hp/hq预设下的编码速度分别为1150/658 FPS
 - 由于驱动尚不完善，本代产品（如RTX 2080）编码速度相比上一代暂时有所下降 (SDK v8.2)

画质比较（SDK v8.2，非最终结果）

	高画质预设(HQ)			高性能预设(HP)		
	RTX 2080	P6000	Delta	RTX 2080	P6000	Delta
BasketballDrive_1920x1080_50.yuv	35.44	35.20	0.24	33.81	33.43	0.39
blue_sky_1920x1080_24.yuv	39.23	38.93	0.30	36.44	36.46	-0.02
Cactus_1920x1080_50.yuv	35.79	35.46	0.33	34.28	34.04	0.24
Kimono_1920x1080_24.yuv	37.84	37.75	0.08	36.15	35.93	0.22
ParkScene_1920x1080_24.yuv	35.32	35.18	0.14	33.55	33.42	0.13
pedestrian_area_1920x1080_24.yuv	39.40	39.26	0.14	37.79	37.52	0.26
sunflower_1920x1080_24.yuv	40.85	40.57	0.28	39.34	38.89	0.45
Tennis_1920x1080_24.yuv	37.77	37.23	0.54	36.10	35.84	0.27
Wisley2_1920x1080_50.yuv	31.89	31.70	0.19	29.57	29.64	-0.07



视频编码实战

- FFmpeg命令行

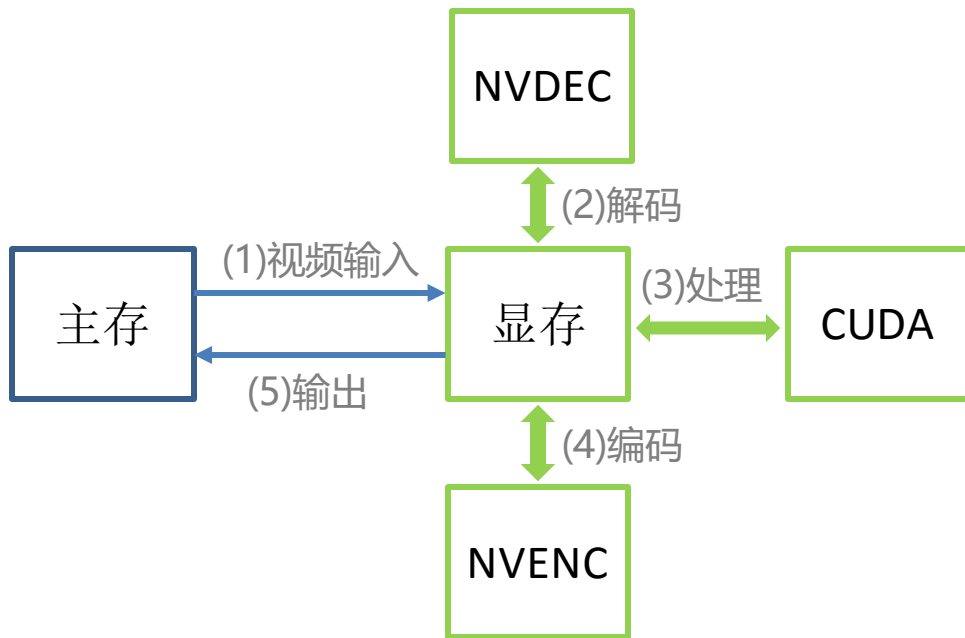
- ffmpeg -i in.h264 -c:v h264_nvenc out.h264
- ffmpeg -hwaccel cuvid -c:v h264_cuvid -i in.h264 -c:v h264_nvenc out.h264

- API

- NVEncode (原名NVENC)
- NvEncLite
- FFmpeg API

```
uint8_t *pHostFrame = new uint8_t[nSize];
NvEncLite enc(cuContext, .._TYPE_CUDA, nWidth, nHeight,
              NV_ENC_BUFFER_FORMAT_NV12);
while (true) {
    int nRead = fread(pHostFrame, 1, nSize, fpSrc);
    vector<vector<uint8_t>> vPacket;
    enc.EncodeHostFrame(nRead==nSize?pHostFrame:NULL,0,vPacket);
    for (vector<uint8_t> &packet : vPacket) {
        fwrite(packet.data(), 1, packet.size(), fpH264);
    }
    if (nRead != nSize) {
        break;
    }
}
```


理想的数据流动





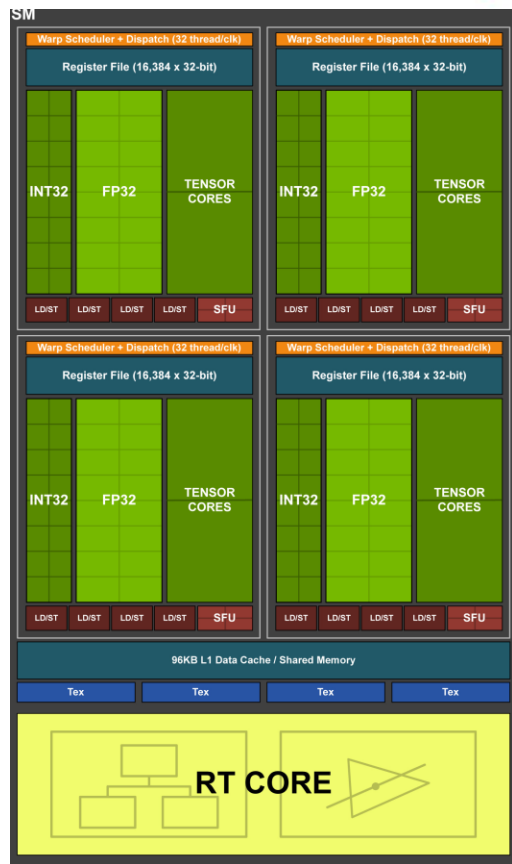
神经网络预测

- MXNet和TensorFlow已经对GPU有良好的支持
 - CPU/GPU可以通过开关切换
 - 由于没有互操作API，无法衔接NVDecode输出的显存数据
- cuDNN和cuBLAS
 - 更基础的计算API
 - 构建了MXNet与TensorFlow的主要GPU功能
 - 接受显存数据为输入
- TensorRT：神经网络预测优化利器
 - 高度优化的GPU计算代码
 - 方便使用Tensor Core (FP16/INT8)
 - 相比效率已经较高的MXNet，能得到3~10倍的加速



Turing的优势： Tensor Core

- GPU的计算单元： Streaming Multiprocessor (SM)
- Tensor Core
 - V100上首发； Turing高端型号全系装备
 - 功能： 单个clock完成4x4矩阵乘加 ($D=A*B+C$)
- 计算能力比较
 - fp16 by Tensor Cores per clock per SM: $(4 \times 4 \times 4) \times 2 \times 8 = 1024$
 - $1.515 \text{ (GHz)} \times 46 \text{ (sm)} \times 1024 = 71362.56 \text{ GFlops}$
 - fp32 by fp32 pipeline per clock per SM: $2 \times 64 = 128$
 - $1.515 \text{ (GHz)} \times 46 \text{ (sm)} \times 128 = 8920.32 \text{ GFlops}$
 - 对比1080 Ti: 无fp16; fp32 10609 GFlops



如何评价Turing GPU?

- 有创新，有舍弃
 - fp32并没有大幅度提高（同级GPU提升了15%~20%）
 - 大量的芯片面积用在了其他功能上
- 更快的解码速度
 - 适用于Tesla/Quadro
 - RTX 2080的解码性能与上代基本持平
- 更高的编码质量
 - Tesla/Quadro无并发路数限制
 - RTX 2080所在整个主机限2路
- 易用高效的Tensor Core加速
 - RTX 2080 AI计算能力(fp16)约为GTX 1080 Ti(fp32)的7倍



Thank you

