CPP-Summit 2019

全球C++软件技术大会

C++ Development Technology Summit

Boolan

高端IT互联网教育平台

关注“博览Boolan”服务号
发现更多 会议·课程·活动

GPU高性能应用开发

CPP-Summit 2019

Ye，Joey

开放智能实验室联合创始人

议程

1 **前言**

程序员的工作

2 **GPU技术**

GPU相关技术介绍

3 **GPU高性能应用案例**

GPU应用性能优化案例

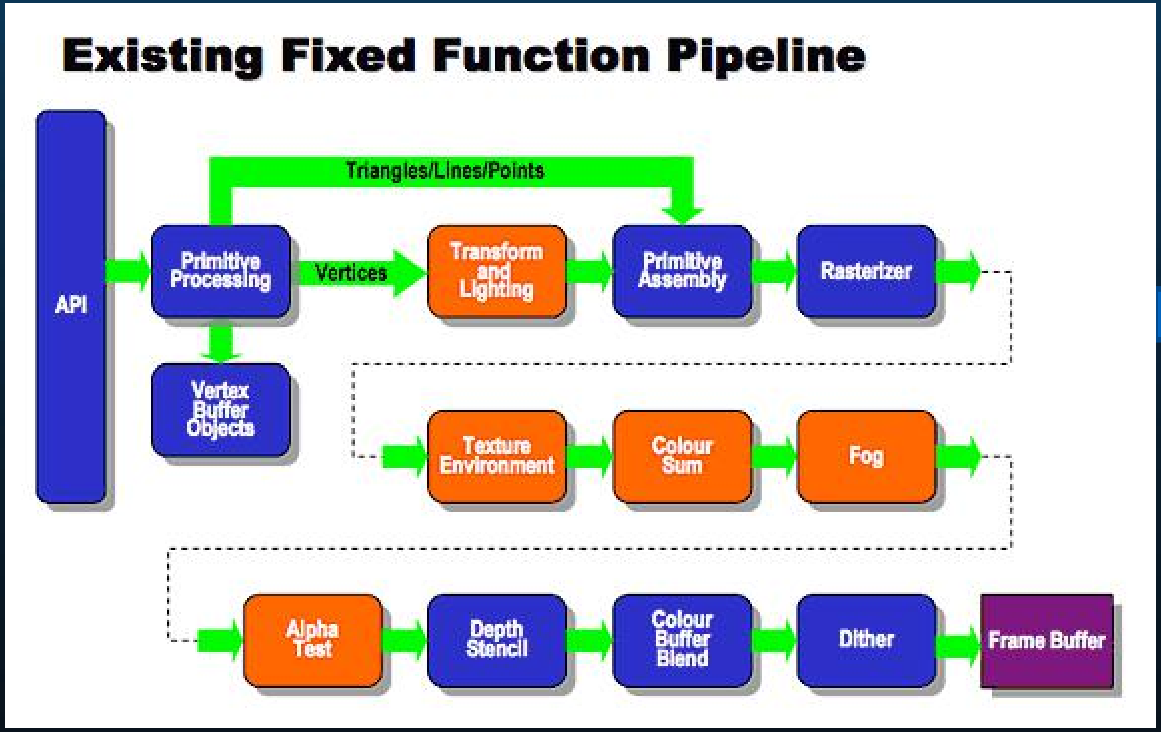
前言

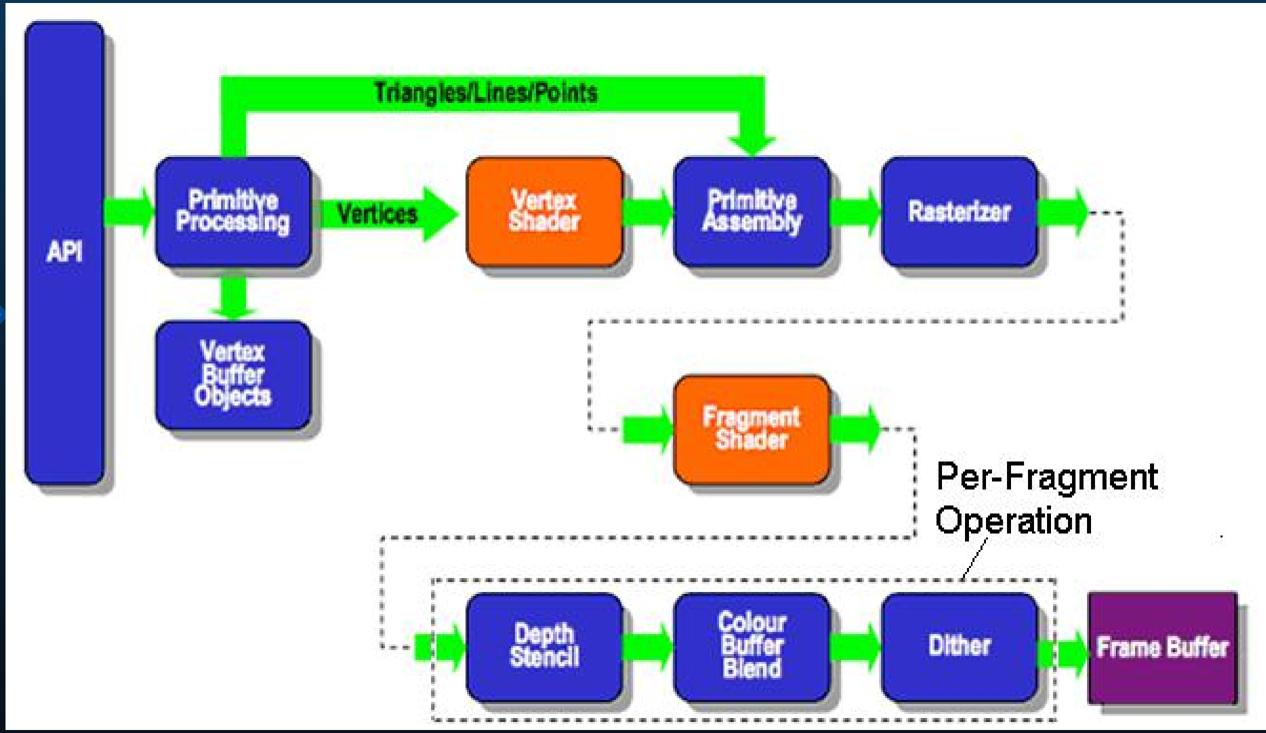
- 调试
- 调优

01

GPU技术

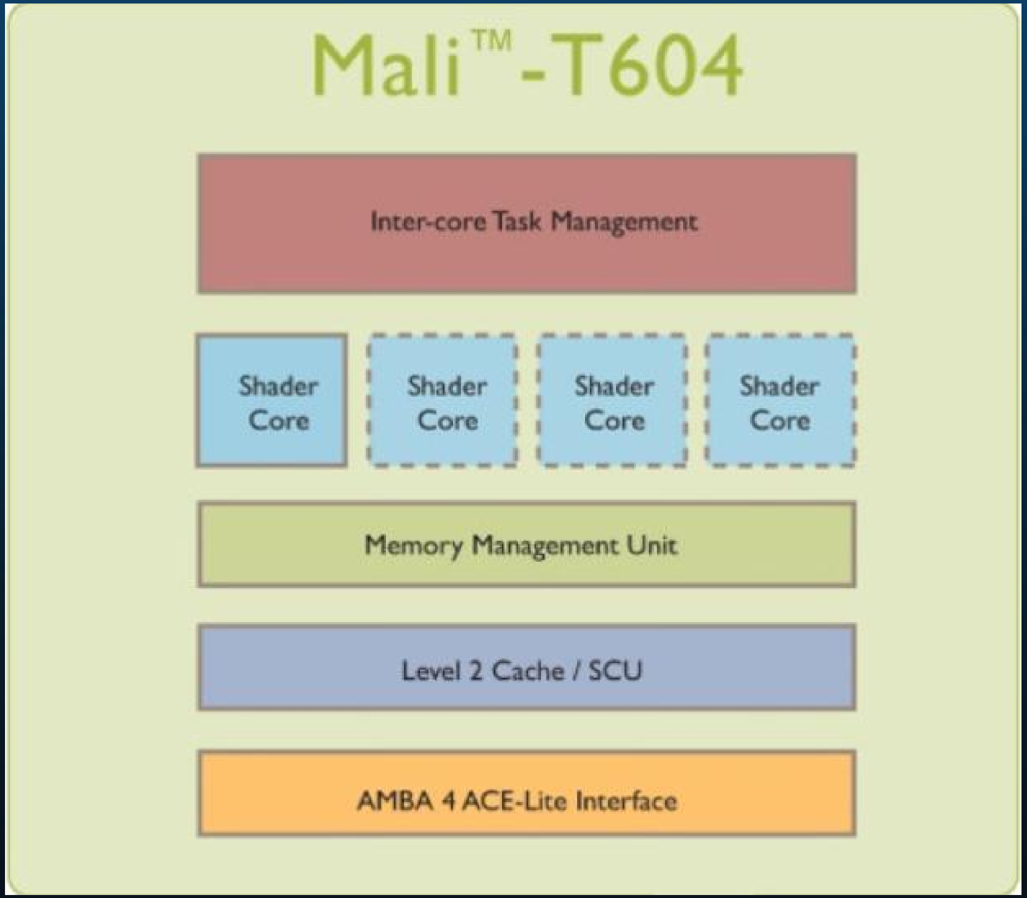
GPU流水线


Existing Fixed Function Pipeline
Triangles/Lines/Points
API
Primitive Processing
Vertices
Transform and Lighting
Primitive Assembly
Rasterizer
Vertex Buffer Objects
Texture Environment
Colour Sum
Fog
Alpha Test
Depth Stencil
Colour Buffer Blend
Dither
Frame Buffer


Triangles/Lines/Points
API
Primitive Processing
Vertices
Vertex Shader
Primitive Assembly
Rasterizer
Vertex Buffer Objects
Fragment Shader
Per-Fragment Operation
Depth Stencil
Colour Buffer Blend
Dither
Frame Buffer

GPU架构

- 256 线程/核
- 支持不同的线程运行相同内核
- 每个线程有自己独立的程序计数器
- 每个线程都有自己的寄存器
- 每个线程有自己的堆栈指针和私有堆栈
- 共享只读寄存器用于内核参数
- 拥有数据缓存



GPU和CPU对比

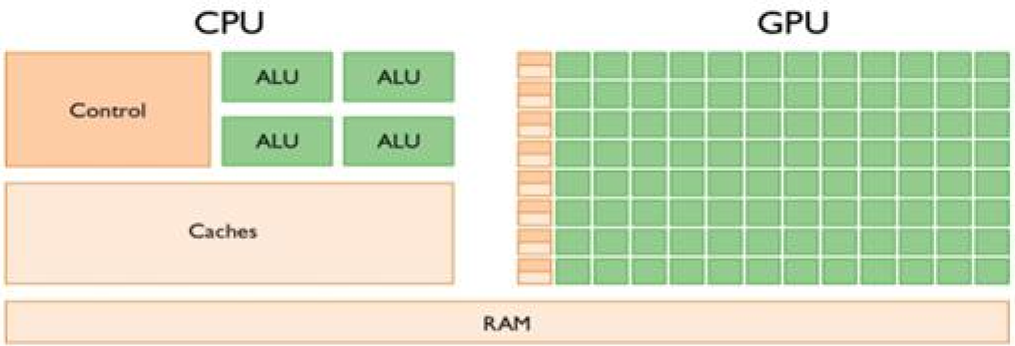
What is GPU Compute

The GPU is now programmable through C-like languages and APIs such as OpenCL™ and Android™ RenderScript

Operating System and most application processing continue to reside on the CPU and can be accelerated through multi-core

The GPU enables cost effective, efficient, and high performance floating point and parallel computation

The GPU can be used as a computational accelerator or as a companion processor



Use cases offloaded to the GPU can include:

- Traditional 2D/3D graphics
- Advanced image processing
- Acceleration/complement of ISP functionality
- Offload of video codec functional blocks
- Acceleration of physics computation

GPU Compute Definition

The use of the GPU for offload and acceleration of non graphical computational tasks

异构计算

- 所有的处理器都是平等
- 公用地址空间
- 处理器间共享指针
- 共享虚拟内存缓存一致性
- CPU和GPU可以创建新的任务
- http://www.hsafoundation.com/

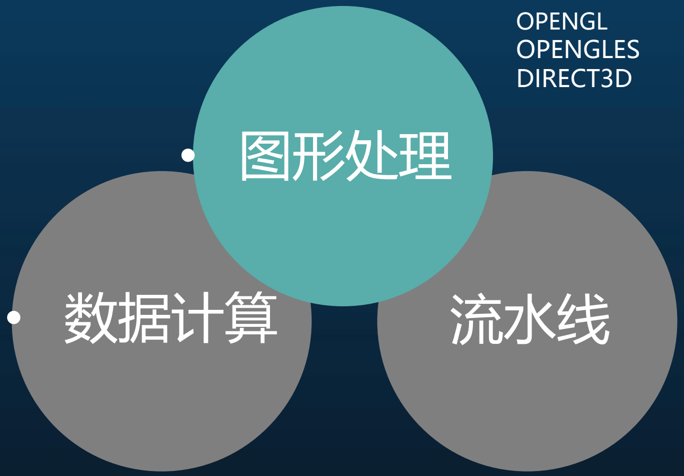
GPU编程

图形处理

OPENGL
OPENGLES
DIRECT3D

数据计算

CUDA
OPENCL
RENDERSCRIPT



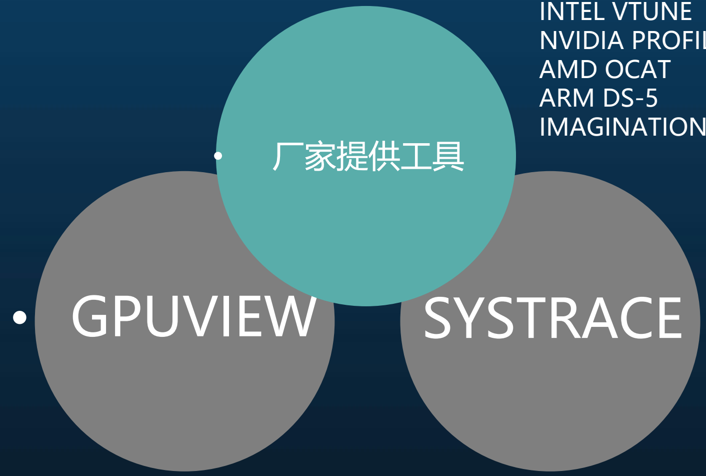
流水线

VULKAN

影响GPU性能主要因素

- 多核
- 工作频率
- 内存带宽
- 数据缓存
- 上下文切换
- 指令优化

GPU性能工具



厂家提供工具

- INTEL VTUNE
- NVIDIA PROFILER
- AMD OCAT
- ARM DS-5
- IMAGINATION PVRTUNE

ARM计算机视觉和深度学习计算库

基于SIMD技术面向ARM CPU/GPU的计算机视觉/深度学习计算库

- 基本算术、数学和二元运算符函数
- 颜色处理（转换，通道提取等）
- 卷积滤波器（Sobel，高斯等）
- Canny边缘，Harris角点，光流等
- 卷积神经网络构建块（激活，卷积，全连接，局部连接，归一化，池化，Softmax）
- 拉普拉斯金字塔
- 面向梯度直方图（HOG）
- 支持向量机（SVM）
- 半单精度的通用矩阵相乘（ H / SGEMM ）

ARM计算机视觉和深度学习计算库

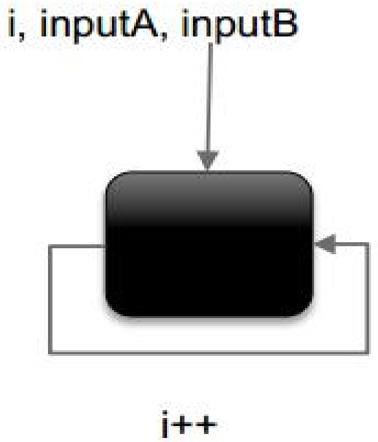
应用场景
- 360度摄像机全景拼接
- 智能相机
- 虚拟与增强现实
- 图像分割
- 特征检测与提取
- 图像处理
- 跟踪
- 深度计算
- 基于机器学习的算法

02

GPU高性能应用案例

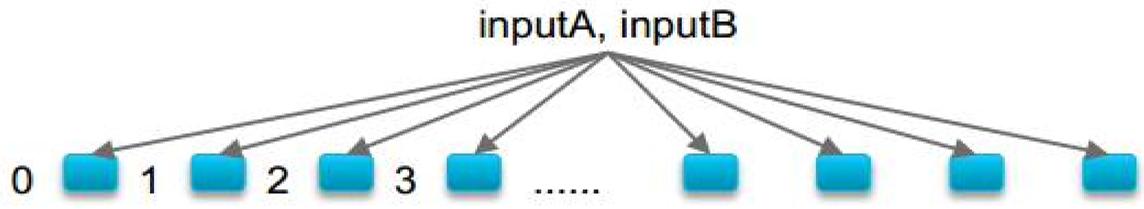
案例1—OPENCL SCALAR ADD

```
for (int i = 0; i < arraySize; i++)
{
    output[i] =
                inputA[i] + inputB[i];
}
```

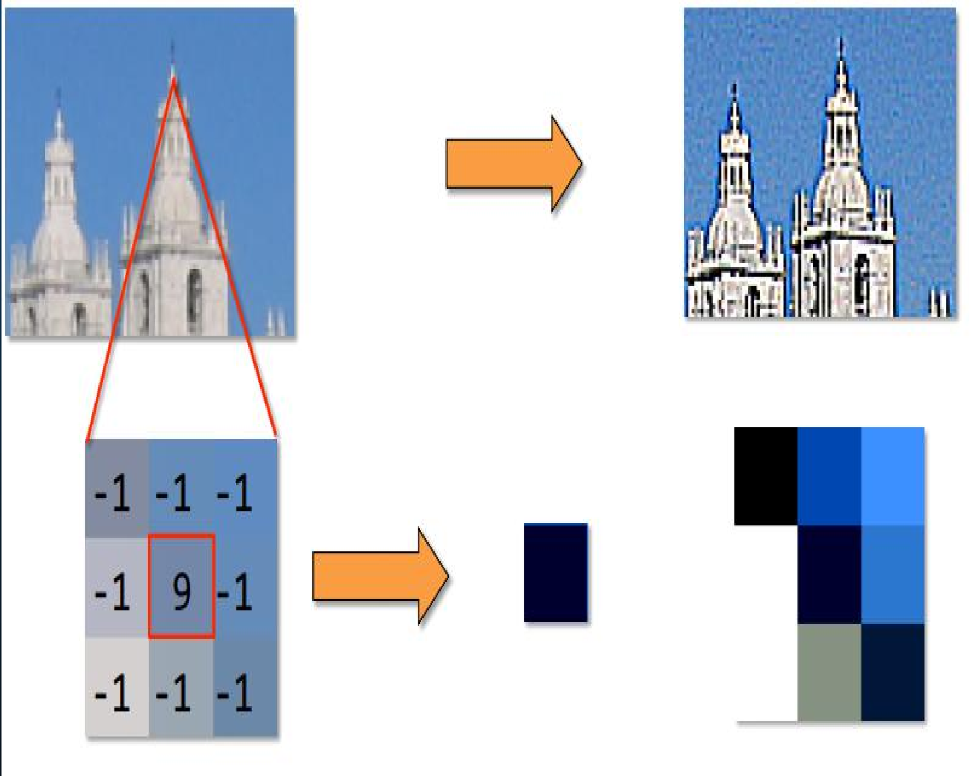


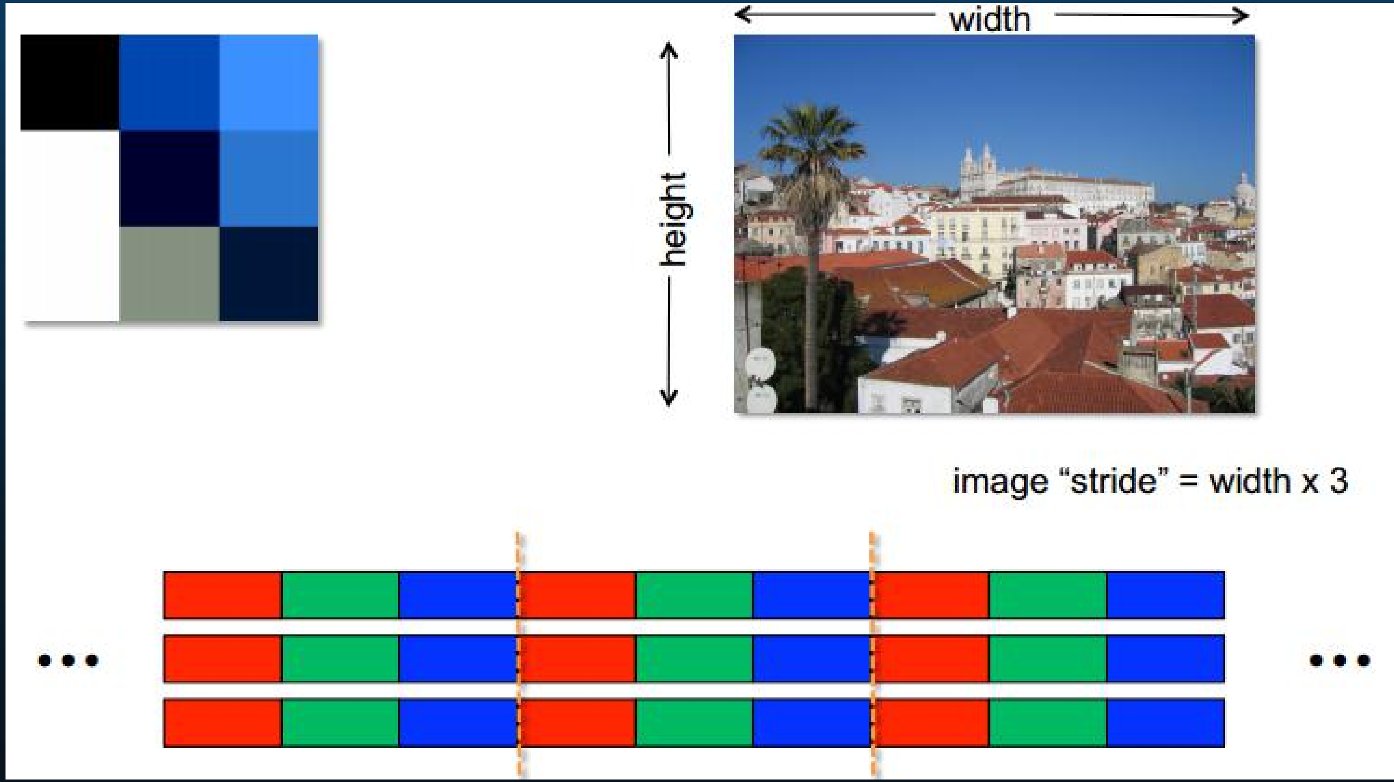

```
__kernel void kernel_name(__global int* inputA,
                          __global int* inputB,
                          __global int* output)
{
    int  i = get_global_id(0);
    output[i] = inputA[i] + inputB[i];
}

clEnqueueNDRangeKernel(..., kernel, ..., arraySize, ...)
```



实例2—拉普拉斯算子优化





实例2—拉普拉斯算子优化

•

```
__kernel void math(__global unsigned char *pdst, __global unsigned char *psrc, int width, int height)
{
    int y           = get_global_id(0);
    int x           = get_global_id(1);
    int w           = width;
    int h           = height;
    int ind         = 0;
    int xBoundary = w - 2;
    int yBoundary = h - 2;

    if (x >= xBoundary || y >= yBoundary)
    {
        ind             = 3 * (x + w * y);
        pdst[ind]       = psrc[ind];
        pdst[ind + 1] = psrc[ind + 1];
        pdst[ind + 2] = psrc[ind + 2];
        return;
    }

    int bColor = 0, gColor = 0, rColor = 0;
    ind          = 3 * (x + w * y);

    bColor = bColor - psrc[ind] - psrc[ind+3] - psrc[ind+6] - psrc[ind+3*w] + psrc[ind+3*(1+w)] * 9 -
             psrc[ind+3*(2+w)]- psrc[ind+3*2*w]- psrc[ind+3*(1+2*w)]- psrc[ind+3*(2+2*w)];
    gColor = gColor - psrc[ind+1] - psrc[ind+4] - psrc[ind+7] - psrc[ind+3*w+1] + psrc[ind+3*(1+w)+1] * 9 -
             psrc[ind+3*(2+w)+1]- psrc[ind+3*2*w+1]- psrc[ind+3*(1+2*w)+1]- psrc[ind+3*(2+2*w)+1];
    rColor = rColor - psrc[ind+2] - psrc[ind+5] - psrc[ind+8] - psrc[ind+3*w+2] + psrc[ind+3*(1+w)+2] * 9 -
             psrc[ind+3*(2+w)+2]- psrc[ind+3*2*w+2]- psrc[ind+3*(1+2*w)+2]- psrc[ind+3*(2+2*w)+2];

    unsigned char blue  = (unsigned char)MAX(MIN(bColor, 255), 0);
    unsigned char green = (unsigned char)MAX(MIN(gColor, 255), 0);
    unsigned char red   = (unsigned char)MAX(MIN(rColor, 255), 0);
    ind             = 3 * (x + 1 + w * (y + 1));
    pdst[ind]       = blue;
    pdst[ind + 1] = green;
    pdst[ind + 2] = red;
}
```

实例2—拉普拉斯算子优化

			Vectorize	Synth. loads	Shorts	4 Pixels	8 Pixels
Image	Pixels	Original	Opt 1	Opt 2	Opt 3	Opt 4	Opt 5
768 x 432	331,776	0.0107	x1.4	x1.4	x1.5	x1.6	x1.2
2560 x 1600	4,096,000	0.0850	x4.5	x4.5	x6.2	x5.2	x5.6
2048 x 2048	4,194,304	0.0865	x1.7	x2.0	x1.9	x5.3	x5.8
5760 x 3240	18,662,400	0.382	x6.0	x6.0	x8.5	x7.2	x8.4
7680 x 4320	33,177,600	0.680	x6.2	x6.3	x9.0	x7.5	x9.1
	Work registers:	8	8+	8	7	6	8+
	ALU cycles:	25.5	22.5	24.5	13.5	14	24
	L/S cycles:	28	13	8	9	6	11

案例3--CHROMIUM

开发环境

- SGX545 @1080p
- HTML5/Javascript + WebGL

设计目标

- 界面更新 @60hz

- HTML CSS层与WebGL 在主线程中共享上下文（Context)，chromium调用eglMakeCurrent()进行上下文切换

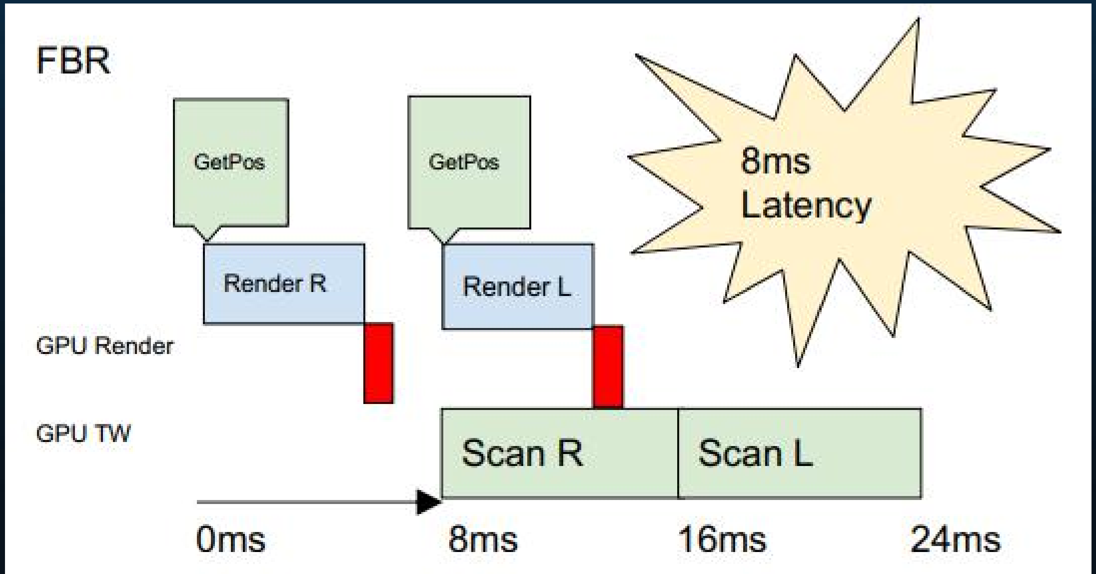
案例4—虚拟现实

硬件环境

- Intel HD Graphics (Cherry Trail)@2K
- 2K屏@60Hz

设计目标

- 总显示延迟 <20ms



Eye Rendering

```
WaitForVBlank()
GetHeadPos()
DrawRightEyeToTexture()
BlitRightEyeToScreen()
eglWaitClient()

GetHeadPos()
DrawLeftEyeToTexture()
BlitLeftEyeToScreen()
eglWaitClient()
```

谢谢