



 JEMO景美

JM11 GPU产品介绍与编程培训

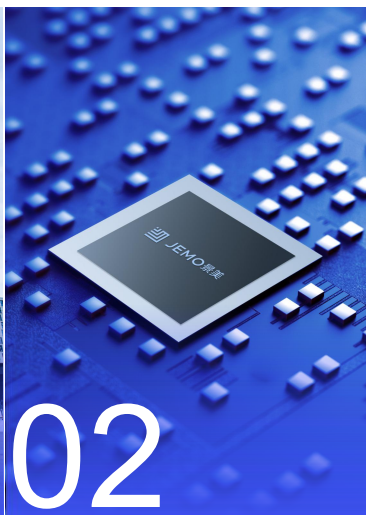
长沙景美集成电路设计有限公司

www.jemoic.com

CONTENTS 目录



公司介绍



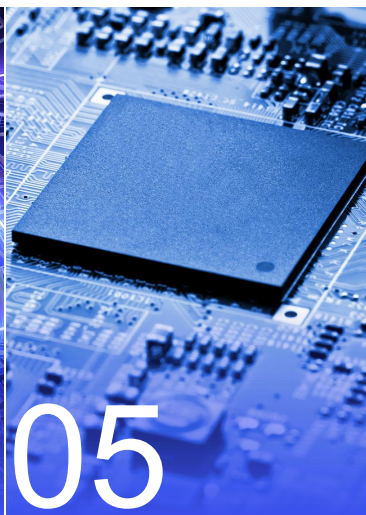
产品介绍



解决方案



驱动安装



应用开发

03

解决方案



场景痛点

数据量庞大、运算复杂、无GPU则常出现加载缓慢
地图缩放、平移时画面延迟、卡顿等问题
严重影响 GIS 系统操作体验与数据分析效率
难以满足高质量三维可视化需求

解决方案

- 利用JM11系列的并行计算能力和高显存带宽，可快速处理、渲染海量GIS数据，高像素填充率使地图画面在缩放、平移过程中保持清晰、流畅，无延迟现象。
- 在三维地理场景构建方面，JM11 精确呈现地形地貌细节，实现逼真光照效果和材质质感，为用户提供沉浸式三维地理分析环境。

应用成效

满足国产化指控应用需求

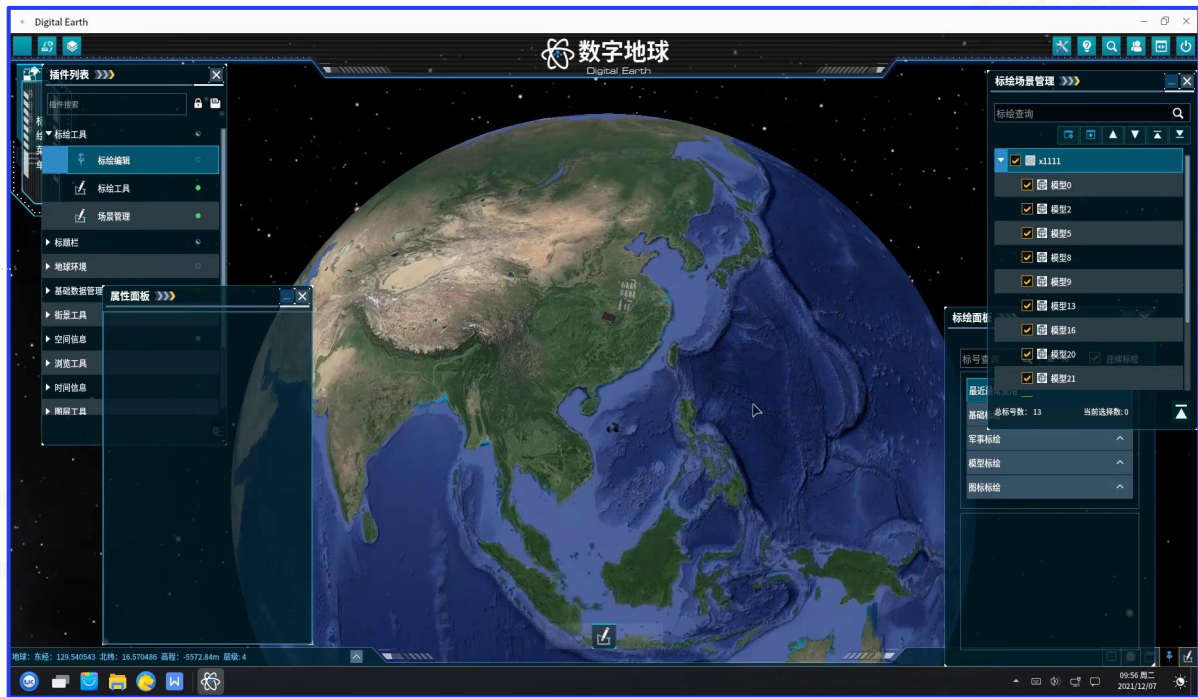
与中科星图、超图、庚图、恒哥等国内GIS厂商完成了适配；

为地理研究、城市规划等工作提供有力支持

在三维地理分析中，科研人员和城市规划者能更直观、准确分析地形地貌特征；

极大提高 GIS 系统使用效率

缩放、平移等地图操作顺滑流畅，可快速定位地理区域；



场景痛点

面对复杂 CAD 文件,易出现模型显示卡顿、线条模糊、图形渲染不及时等问题,导致设计师绘图、修改和查看设计成果时操作不流畅,影响设计效率和质量。部分国产 CAD 软件与国外显卡存在兼容性问题。

解决方案

- 长城D3000主机、麒麟OS加JM1100显卡,提供全国产化图站方案;
- 运行中望CAD软件,600+模型加载,百兆以上模型素材渲染,CPU占用率不到4%。
- 卡顿现象不发生,平移和缩放等操作非常流畅,获得了设计师们的好评。

应用成效

完成国产CAD适配

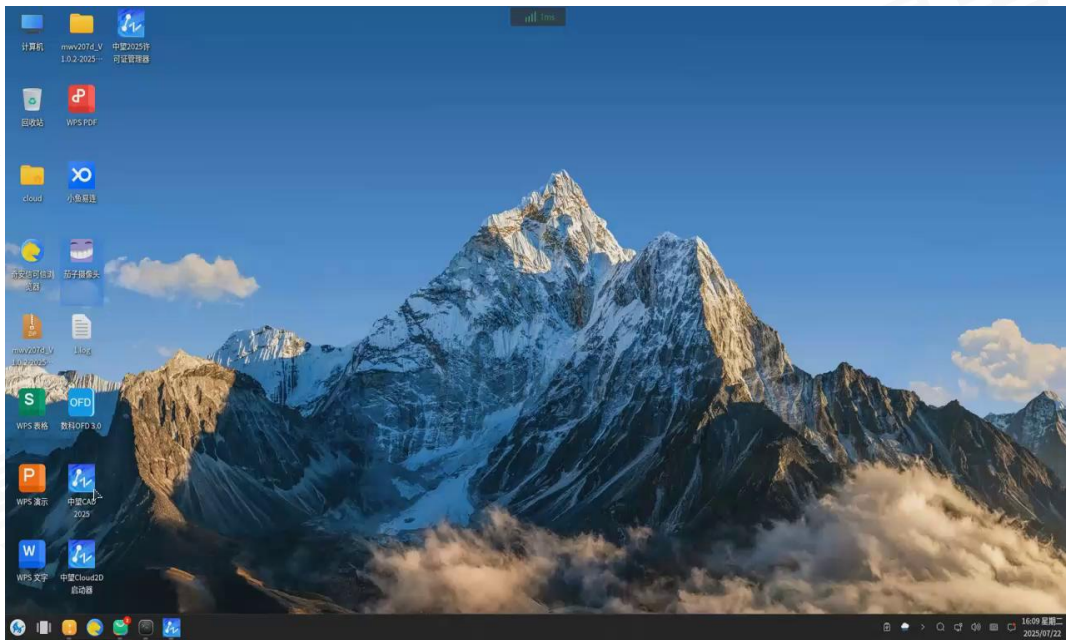
与中望CAD、浩辰CAD等国内厂商合作

首次全国产化显卡环境下运行

实现CAD工业研发设计软件

图形渲染速度提升,大幅缩短设计周期

显卡与国产CAD软件兼容性深度适配,减少异常情况



场景痛点

能源、工业制造等国计民生场景中有大量的数字孪生需求，通过3D可视化的方案，在生产运维、人员定位等场景可以是先更加直观、有效的管理方式。该场景有着强烈的全国产化需求，安全、可靠的长周期服务，是客户的基本要求

解决方案

- 基于C86整机加JM1100打造了全国产化图形工作站，安装麒麟系统，适配国产化3D引擎底座，1080P@60fps无卡顿流畅工作
- 通过数字化交付，实现了动态数据与静态数据融合，实现了安全管理（人员定位）、巡检、设备管理、生产等各业务的数字化应用

应用成效

完成国产3D引擎底座适配

与达美盛、漂视科技的等完成引擎适配

全国产化环境下运行

实现日常运维巡检、人员及设备管理等核心需求

3D渲染流畅而稳定 7x24小时无故障工作

深度优化提升性价比 针对引擎底座和3D场景深度优化





应用成效

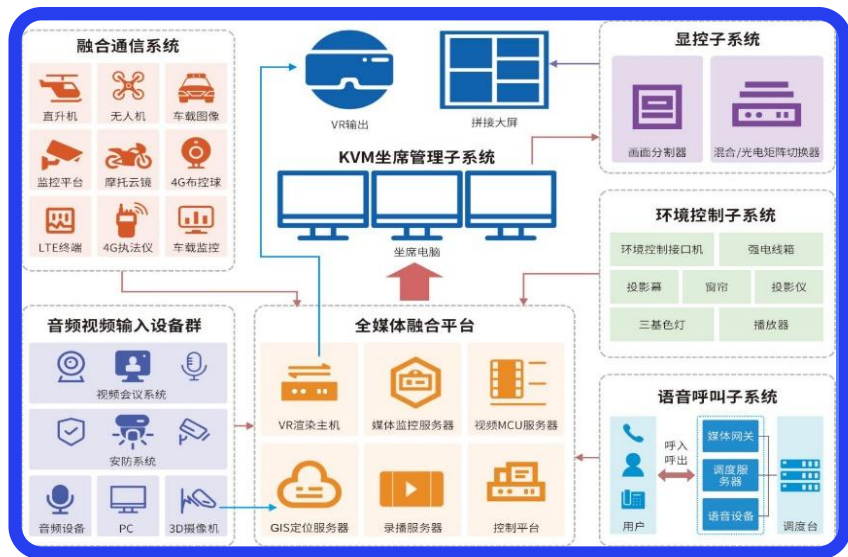
目前已经兼容海康、大华等视频监控软件，支持多路异屏输出
流畅显示在监控屏幕上，解码速度快、丢帧现象极少发生

场景痛点

公共安全视频监控建设联网应用，9路、16路、32路监控视频回看客户端，面对公安系统各级单位的信息化改造，通过云桌面来解决视频监控以及办公电脑面临维护繁琐，视频及其他数据在本地易丢失等问题，多路高清监控视频流时，解码速度慢、丢帧现象严重，无法实时准确显示监控画面，影响监控人员对异常情况的及时发现和处理。

解决方案

- 通过GPU硬件解码功能实现4路4K@60帧超高清视频同时解码播放，16路1080P@60帧高清解码，支持4路和9路以上高清视频同屏播放，并将桌面查看的视频内容投射到大屏
- 利用终端上JM1100的4路HDMI实现4路异显扩展屏幕



应用成效

结合国产操作系统深度开发，具有**高度安全性和完整的自主知识产权与专利技术**；

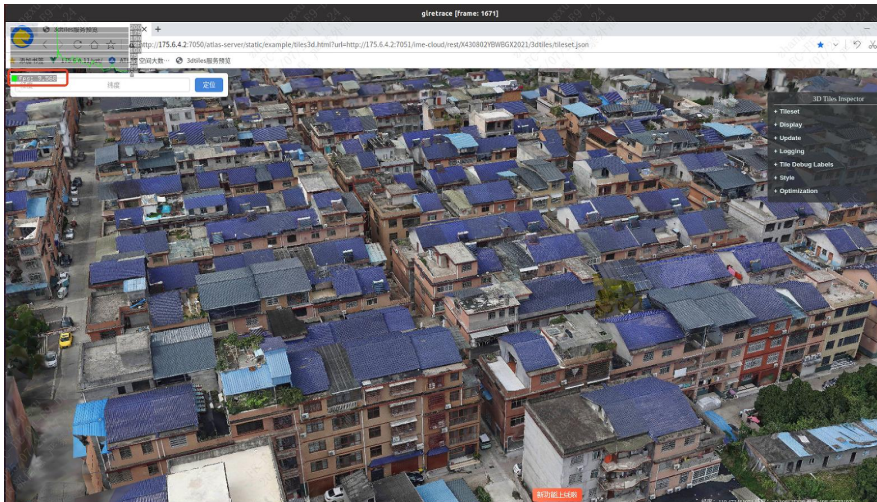
对网内各类视频源设备进行切换管理，实时预览、拖拽上屏、决议下达。

解决方案

景美联合飞腾推出了城市调度大厅多媒体管理系统解决方案中KVM坐席管理子系统，用于导调人员对数据、事件进行判别决策。

KVM坐席管理子系统和音视频输入设备群采用了飞腾D2000、景美JM1100系列GPU显卡并搭载国产操作系统。

实现处理器、GPU、操作系统、基础软件的全面国产化替代



应用成效

- 地图测绘行业全国产化（国产化CPU+国产OS+国产GPU）解决方案落地
- 显卡与测绘行业软件兼容性深度适配，优化测绘软件运行速度

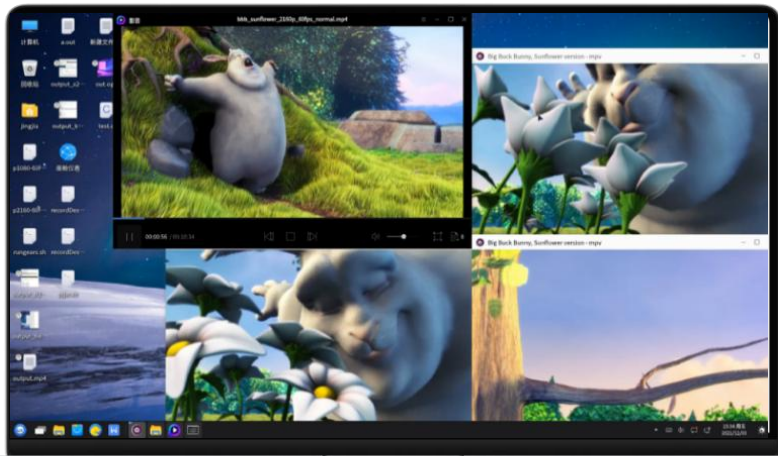
场景特点

三维建模与实景渲染处理无人机航拍影像、激光点云或三维地形模型时，GPU成为核心性能瓶颈：

- **渲染负载**：需实时计算光照、纹理贴图及几何变换，依赖GPU的流处理器数量和核心频率
- **显存容量**：高分辨率影像和点云数据占用显存极大。
- **并行计算加速**：三维重建算法依赖GPU的并行算力提升建模效率。

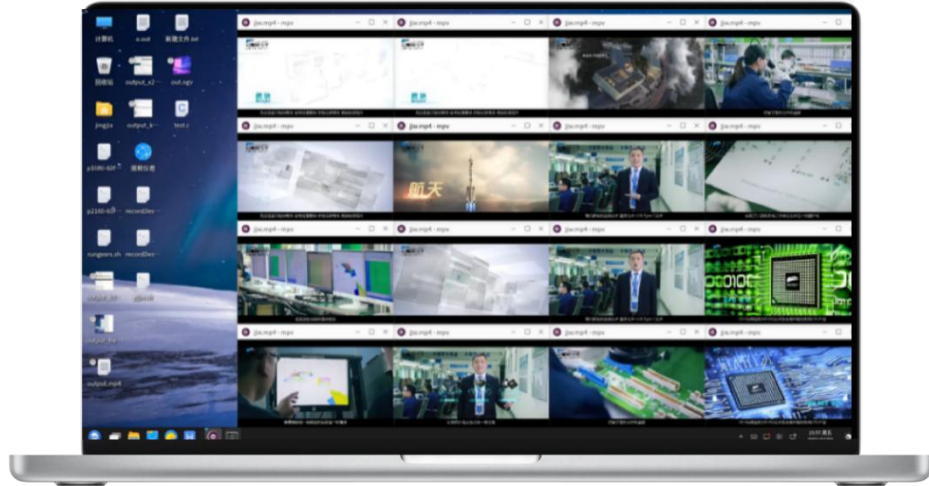
解决方案

- 像素、纹理填充率达38.4GPixel/s可应付大量的三维建模与实景渲染场景
- 16GB显存容量可以存储大量的高分辨率测绘数据
- 1.3TFLOPS的算力支持三维建模算法高效运行



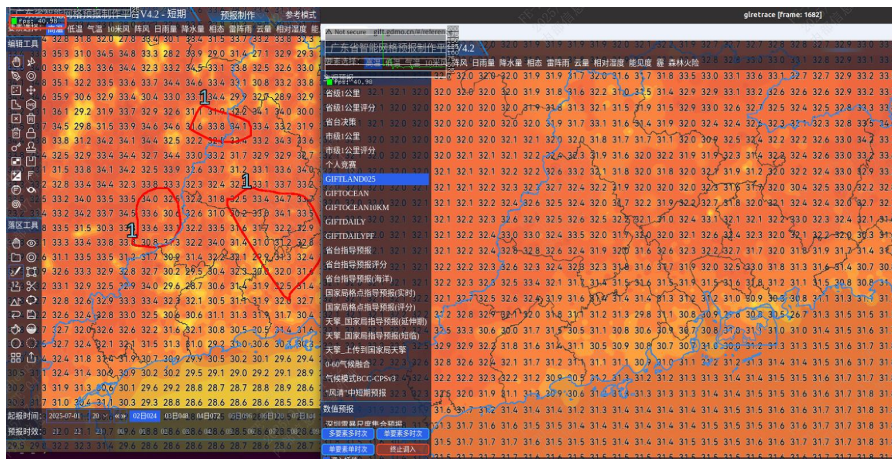
方案介绍

- 基于JM1100国产自主可控GPU显卡产品
- 适配国产CPU芯片及麒麟操作系统，支持飞腾、龙芯等多种国产化CPU和操作系统平台；
- JM1100支持H.265、H.264、VP9等高清视频解码等功能,调用规范VA-API接口实现解码功能。



应用成效

在**国产CPU+国产OS平台**下通过GPU硬件解码功能实现**4路4K@60fps超高清**视频同时解码播放，及实现**16路1080P**视频同时解码播放。



应用成效

- 地图测绘行业全国产化（国产化CPU+国产OS+国产GPU）解决方案落地
- 显卡与气象行业软件兼容性深度适配，优化气象软件运行速度

场景特点

气象数据（如温度场、湿度场、风场矢量，降水数据）需要通过3D模型动态展示,此类场景需要实时渲染数百万多边形，并叠加物理光照效果，对GPU的浮点运算和渲染性能要求极高。

景美方案

- 像素、纹理填充率达38.4GPixel/s可应付大量的气象数据的三维建模渲染场景
- 大显存容量可以存储大量的气象数据，满足海量数据的加载显示

04

驱动安装



JM1100通用显卡驱动版本相关信息

驱动包名称	mwv207d-mesa-dkms
版本号定义	x.y.z-yyyymmdd.hhmm-OS.SERVER-k.k_ARCH
版本示例	mwv207d-mesa-dkms_1.0.9-20251015.0846-openKylin.2.0-6.6_amd64.deb
适配架构	amd64 / arm64 / loongarch64
CPU运算能力	飞腾D2000/D3000、鲲鹏920、龙芯3A5000/6000、Intel、兆芯
GPU	Jingjia MWV207D / JM1100 系列
操作系统:	Kylin V10 SP1 / Ubuntu 20.04 等
内核版本	5.4 / 5.10 等
依赖	dkms、libdrm2、libgbm1、libudev1、mesa-glx、ocl-icd-libopenc1

JM1100通用显卡驱动版本安装与卸载

安装驱动	<code>sudo dpkg -i mww207d-mesa-dkms_1.0.9-20251015.0846-openKylin.2.0-6.6_amd64.deb</code> 安装成功提示: Driver installation succeeded. 需重启生效
查看驱动安装情况	<code>dpkg -l grep mww207d</code>
卸载驱动	<code>sudo dpkg -P mww207d-mesa-dkms</code>
jm-smi工具	<code>jm-smi -q</code> 可查看 GPU 名称、PCIe 信息、显存使用率、温度等, 显示信息如下: =====JMSMI LOG===== Timestamp : Fri Jun 6 17:37:48 2025 Driver Version : 1.0 Attached GPUs : 1 GPU 0000:06:00.0 Product Name : JMI100 . . . Utilization 3D : 0 % 2D : 0 % Memory : 1 % Fus : 0 % Encoder : 0 % Decoder : 0 %

JM1100通用显卡驱动版本安装与卸载

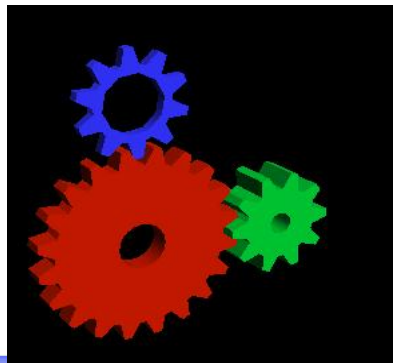
3D 硬解验证状态确
认

1. 终端执行vblank_mode=0 glxgears

```
jingjia@jingjia-System-Product-Name:~$ vblank_mode=0 glxgears
ATTENTION: default value of option vblank_mode overridden by environment.
74850 frames in 5.0 seconds = 14969.929 FPS
75429 frames in 5.0 seconds = 15085.757 FPS
```

2. 另开终端执行jm-smi -q,查询到 jm-smi 的数据, 观察 Utilization/3D 的数据,
预期结果: "Utilization/3D" 数值 > 0

```
Free Space : 155 MB
Utilization
  3D       : 90 %
  2D       : 54 %
  Memory   : 1 %
  Fus      : 0 %
  Encoder  : 0 %
  Decoder  : 0 %
```



JM1100通用显卡驱动版本安装与卸载

常见问题

问题 1：依赖不足导致安装失败

现象：

终端提示 “dpkg: 依赖关系问题使得 mww207d-mesa-dkms 的配置工作不能继续”，如缺失 “dkms”

解决方案：

安装缺失依赖：sudo apt install 缺失包名（如sudo apt install dkms）；

若未自动续装，卸载后重新安装：sudo dpkg -P mww207d-mesa-dkms && sudo dpkg -i 驱动包.deb

问题 2：内核版本不匹配

现象：

终端提示 “Kernel Version Checking ... ERROR, Built for kernel 5.10, current is 5.4”

解决方案：

(1) 查看当前内核：uname -a（如输出 “5.4.18-110-generic”）；

(2) 更换对应内核的驱动包（如内核 5.4 对应 “-5.4_” 后缀的驱动包）；

(3) 重新安装：sudo dpkg -P mww207d-mesa-dkms && sudo dpkg -i 对应驱动包.deb

05

应用开发



▶ 核心前提:

- (1) JM1100 支持 OpenCL3.0, 编程接口与通用 OpenCL 无差异;
- (2) 在跑应用之前需要指定GPU为JM1100显卡: `export RUSTICL_ENABLE=mwv207d`

▶ 环境检查步骤:

依赖确认: `dpkg -l | grep ocl-icd-libopencl`, 确保已安装opencl

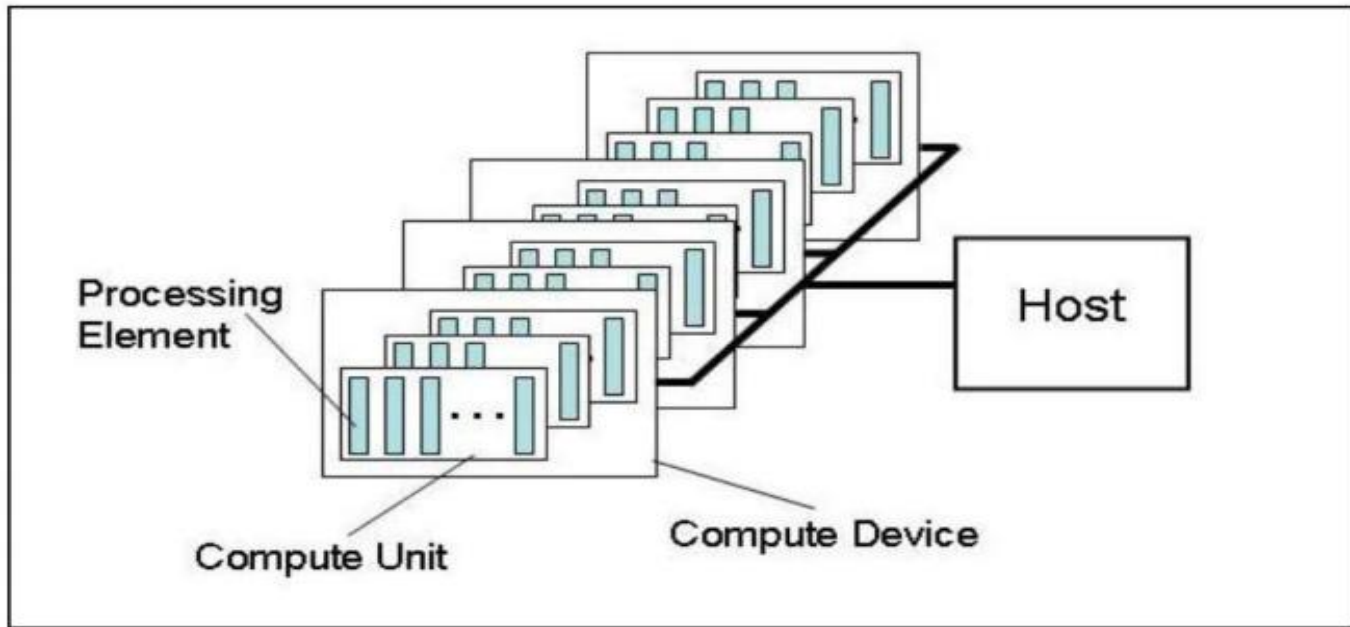
平台识别:

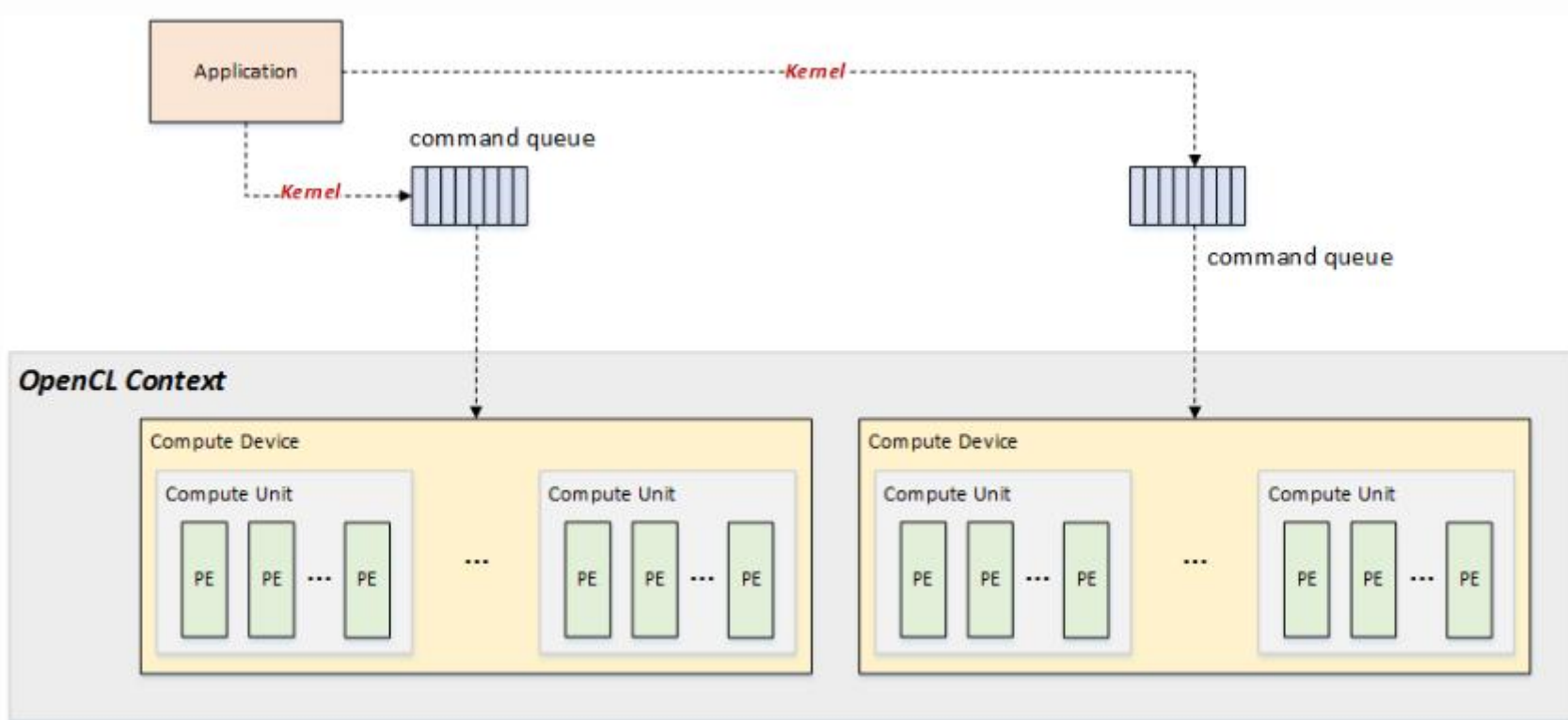
clinfo: 确认 “Device Name” 为: MWV207D (JingjiaMicro Corporation);

确认 “Device Vendor” 为: MWV207D (Vendor-id: 0x0731)

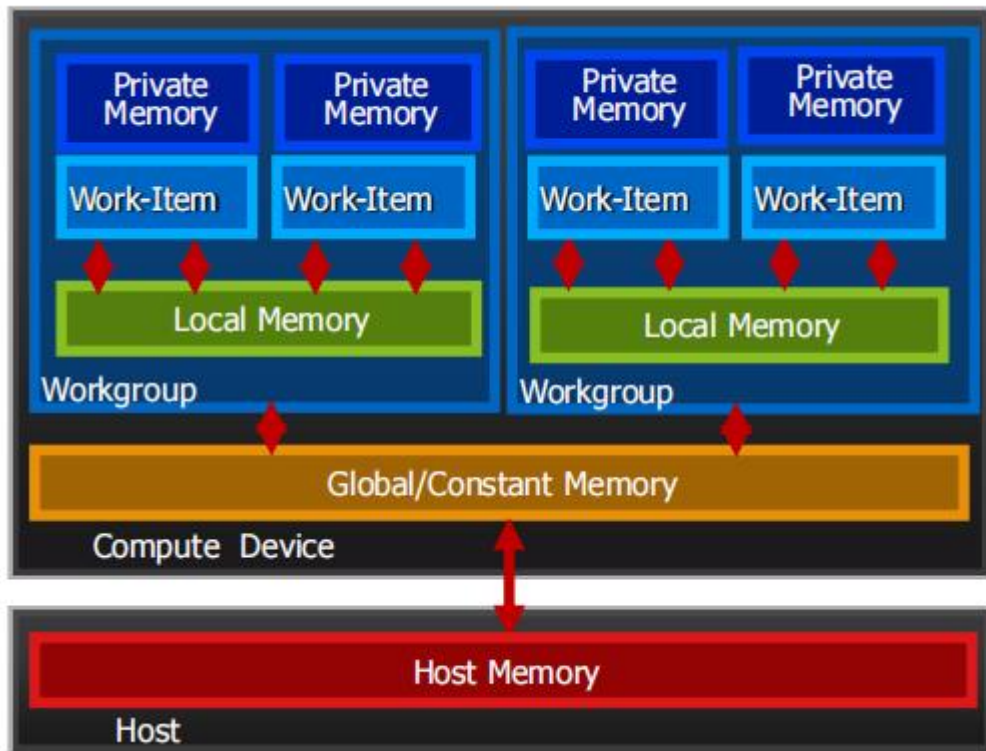
```
jingjia@jingjia-System-Product-Name:~$  
jingjia@jingjia-System-Product-Name:~$  
jingjia@jingjia-System-Product-Name:~$ export RUSTICL_ENABLE=mwv207d  
jingjia@jingjia-System-Product-Name:~$ clinfo  
Number of platforms                                1  
Platform Name                                       rusticl  
Platform Vendor                                    Mesa/X.org  
Platform Version                                    OpenCL 3.0  
Platform Profile                                    FULL_PROFILE  
Platform Extensions                                cl_khr_byte_addressable_store cl_khr_  
er_wrap_decoration cl_khr_suggested_local_work_size  
Platform Host timer resolution                     1ns  
Platform Extensions function suffix               MESA  
  
Platform Name                                       rusticl  
Number of devices                                  1  
Device Name                                         MWV207D (JingjiaMicro Corporation)  
Device Vendor                                       MWV207D (Vendor-id: 0x0731)  
Device Vendor ID                                   0x731  
Device Version                                     OpenCL 3.0  
Driver Version                                     24.1.2 (git-5d155489a5)  
Device OpenCL C Version                           OpenCL C 1.2  
Device Type                                         GPU  
Device Profile                                     EMBEDDED_PROFILE
```

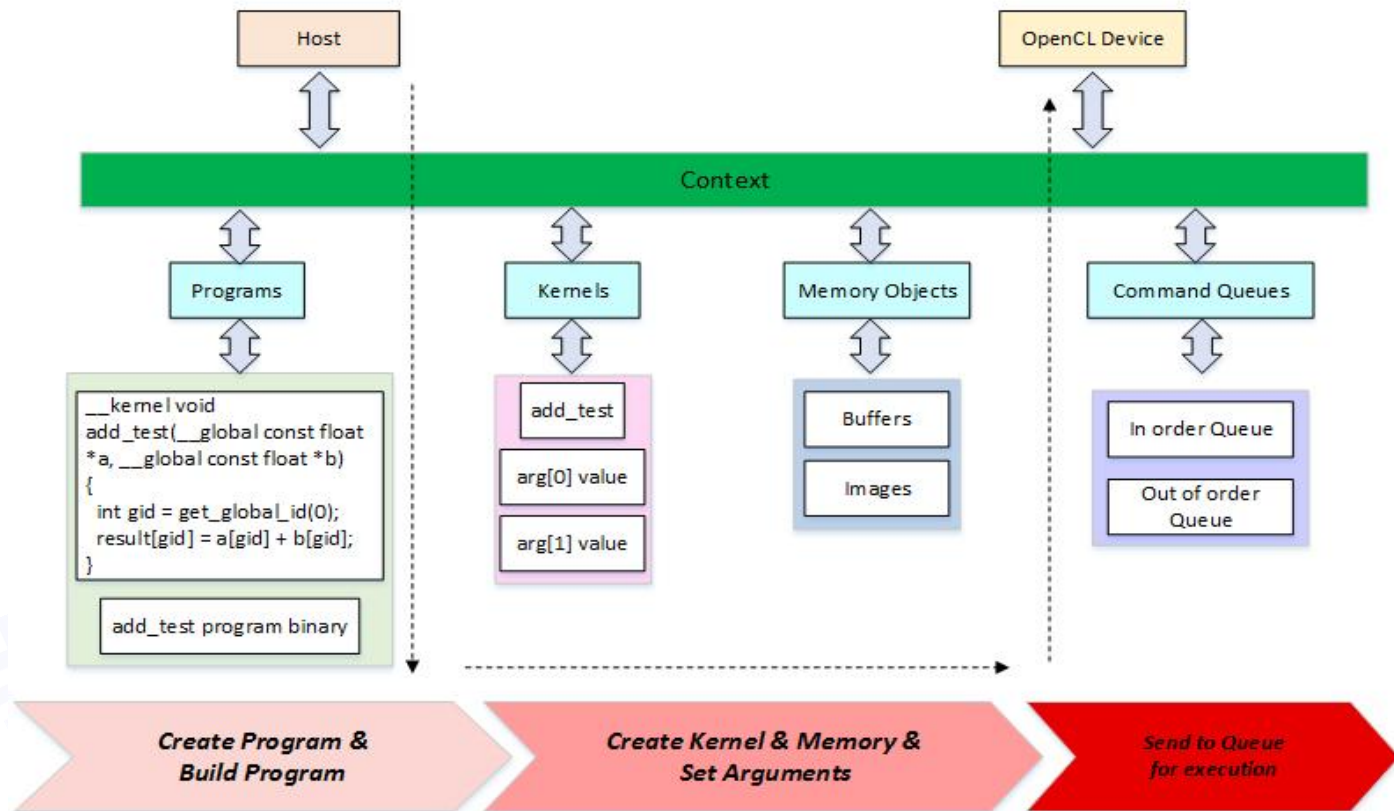
- 平台模型由一个Host连接一个或多个OpenCL Devices组成
 - OpenCL Device, 可以划分成一个或多个计算单元Compute Unit (CU)
 - CU可以进一步划分成一个或多个处理单元Processing Unit (PE)





- work-item->work-group->NDRange
- Private Memory
 - work-item私用
- Local Memory
 - work-group组内共享
- Global/Constant Memory
 - 所有workgroups可见
- Host Memory
 - CPU侧





标准 OpenCL 编程流程

- ▶ 步骤 1: 平台与设备选择
- ▶ 步骤 2: 创建上下文与命令队列
- ▶ 步骤 3: 内核编译与执行
- ▶ 步骤 4: 结果读取与资源释放

```
// 获取平台
err = clGetPlatformIDs(1, &platform, NULL);
CHECK_ERROR(err);

// 优先获取GPU设备
cl_device_type device_type = CL_DEVICE_TYPE_GPU;
err = clGetDeviceIDs(platform, device_type, 1, &device, NULL);
if (err != CL_SUCCESS) {
    printf("未找到GPU设备, 尝试使用CPU...\n");
    device_type = CL_DEVICE_TYPE_CPU;
    err = clGetDeviceIDs(platform, device_type, 1, &device, NULL);
    CHECK_ERROR(err);
}

// 创建上下文、命令队列等(后续步骤与之前一致)
context = clCreateContext(NULL, 1, &device, NULL, NULL, &err);
CHECK_ERROR(err);

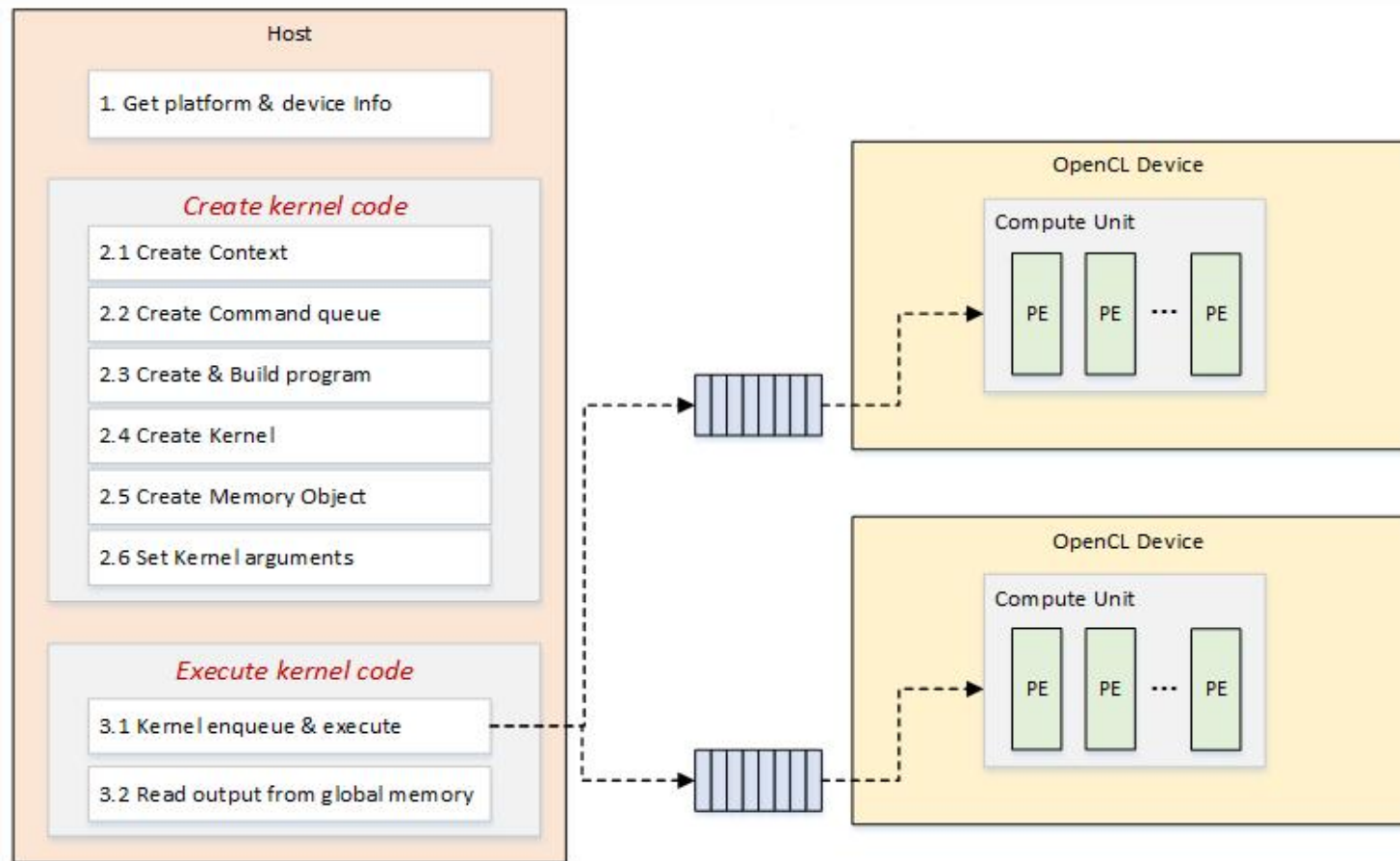
queue = clCreateCommandQueue(context, device, 0, &err);
CHECK_ERROR(err);

program = clCreateProgramWithSource(context, 1, &kernelSource, NULL, &err);
CHECK_ERROR(err);

// 编译内核(带错误日志)
err = clBuildProgram(program, 1, &device, NULL, NULL, NULL);
if (err != CL_SUCCESS) {
    size_t log_size;
    clGetProgramBuildInfo(program, device, CL_PROGRAM_BUILD_LOG, 0, NULL, &log_size);
    char *log = (char*)malloc(log_size);
    clGetProgramBuildInfo(program, device, CL_PROGRAM_BUILD_LOG, log_size, log, NULL);
    fprintf(stderr, "内核编译失败:\n%s\n", log);
    free(log);
    exit(1);
}

kernel = clCreateKernel(program, "vecAdd", &err);
CHECK_ERROR(err);

// 创建缓冲区
d_a = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, bytes, h_a, &err);
d_b = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, bytes, h_b, &err);
d_c = clCreateBuffer(context, CL_MEM_WRITE_ONLY, bytes, NULL, &err);
CHECK_ERROR(err);
```



1. 代码示例

```
_kernel void vec_add(__global float* a, __global float* b, __global float* c) {  
    int id = get_global_id(0);  
    c[id] = a[id] + b[id];  
}
```

2. 编译与执行

```
gcc test_opengl.c -lOpenCL -o test_opengl  
export RUSTICL_ENABLE=mvv207d  
./test_opengl
```

致谢



让中国 “芯” 照亮未来之路

THANKS! 谢谢观看



长沙景美集成电路设计有限公司

www.jemoic.com

全国服务热线: 400 963 9339