乘影开源GPGPU软件工具链介绍

清华大学-软件工程师
孔荔

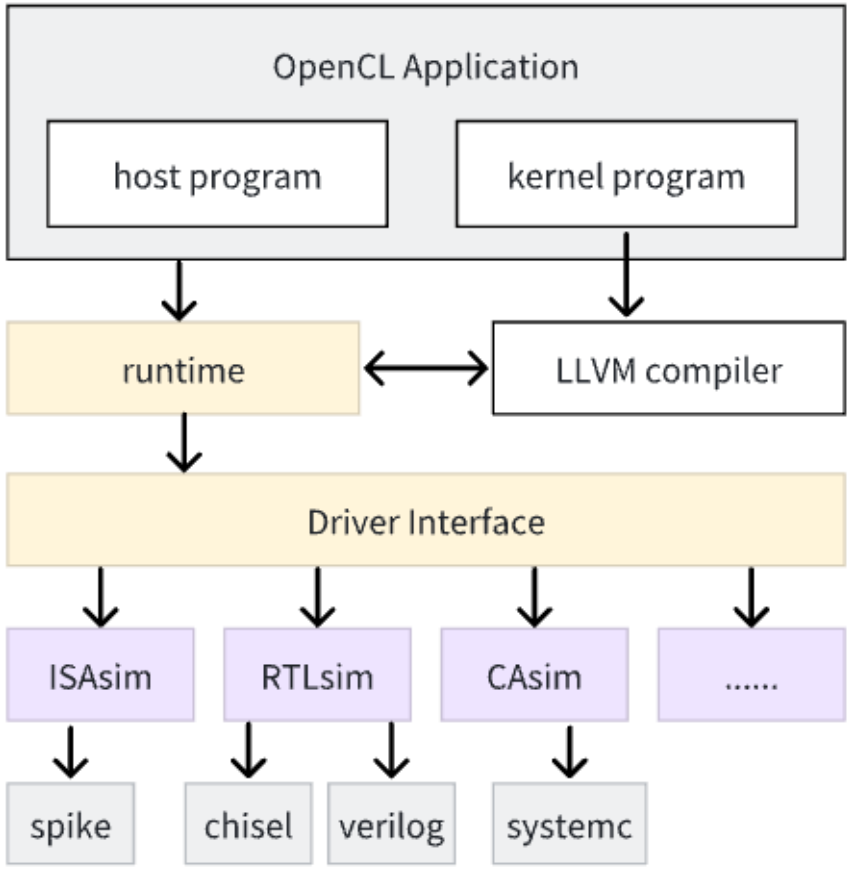
OpenGPGPU
乘影

内容

- 开源GPGPU — 软件工具链设计
  - 整体架构设计
  - 运行时环境设计
  - 驱动程序设计
  - 测试套
  - 搭建与使用
- 开源GPGPU — 开源社区

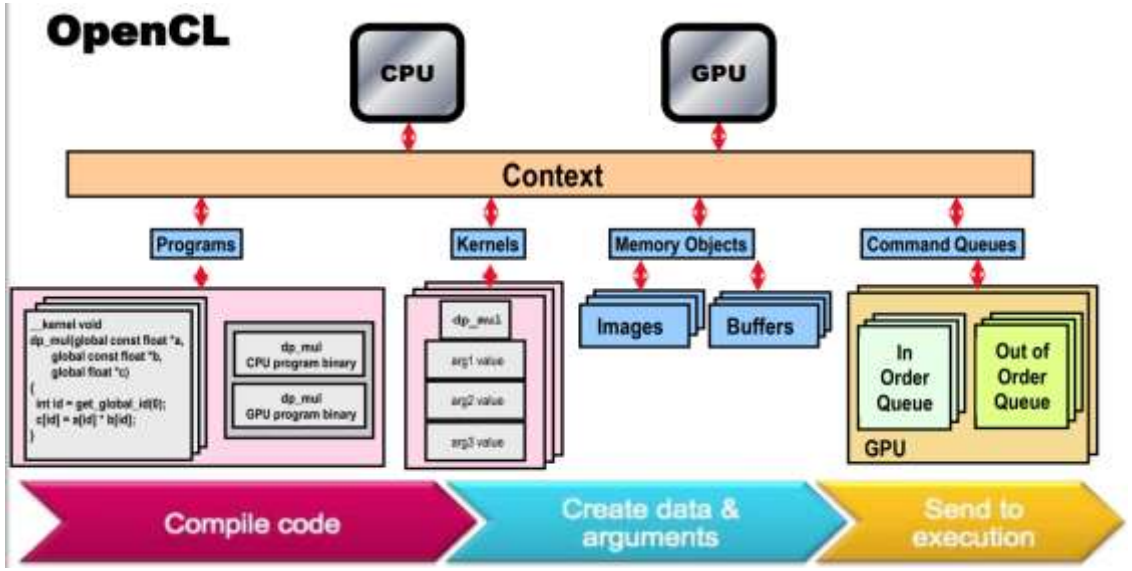
软件工具链 — 整体架构设计

- 实现支持硬件验证和仿真的完整工具链



开源GPGPU的软件工具链整体架构

- 运行时环境（Runtime）
  - 实现OpenCL API接口
  - 编译、运行内核程序
  - 生成metadta
  - .....
- 设备驱动程序（Driver）
  - 定义通用接口，兼容不同类型的设备
  - 实现对GPGPU硬件资源的控制



*OpenCL Overview - The Khronos Group Inc

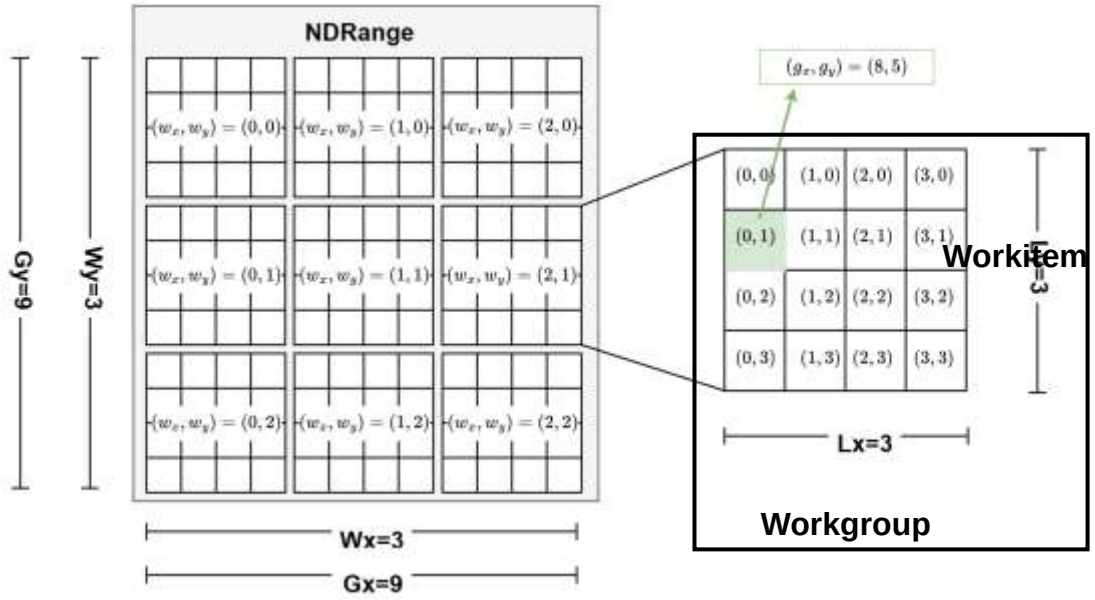
Runtime: GPGPU编程模型映射

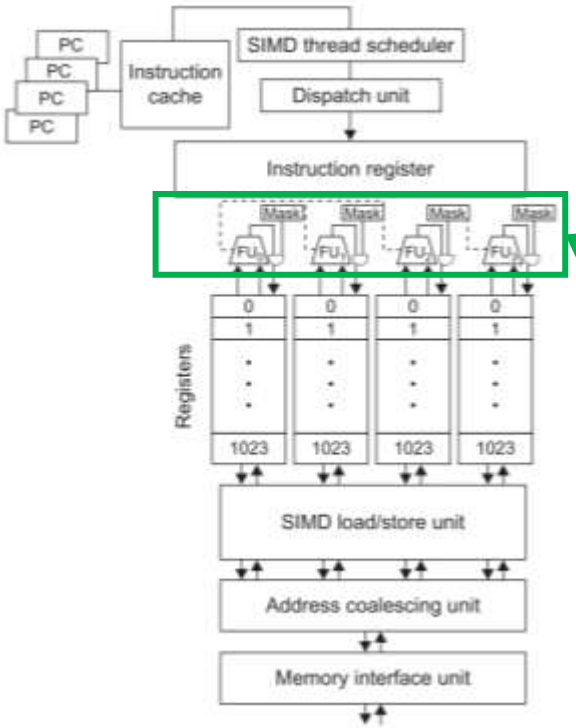
- GPGPU硬件需要映射到OpenCL编程模型

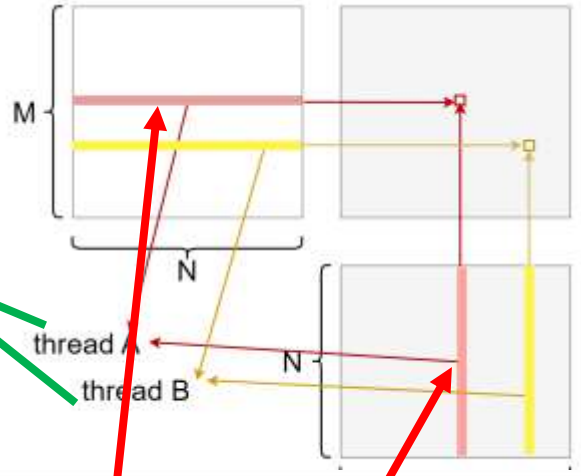
NDRange —— Kernel

Workgroup —— CTA/Block

Workitem —— Thread








```
__kernel void matrix_mult(
    __global float* A,
    __global float* B,
    __global float* C
) {
    int i = get_global_id(0);
    int j = get_global_id(1);

    float sum = 0.0f;
    for (int k = 0; k < 4; k++) {
        sum += A[i * 4 + k] * B[k * 4 + j];
    }
    C[i * 4 + j] = sum;
}
```

OpenCL中一个矩阵乘法的例子

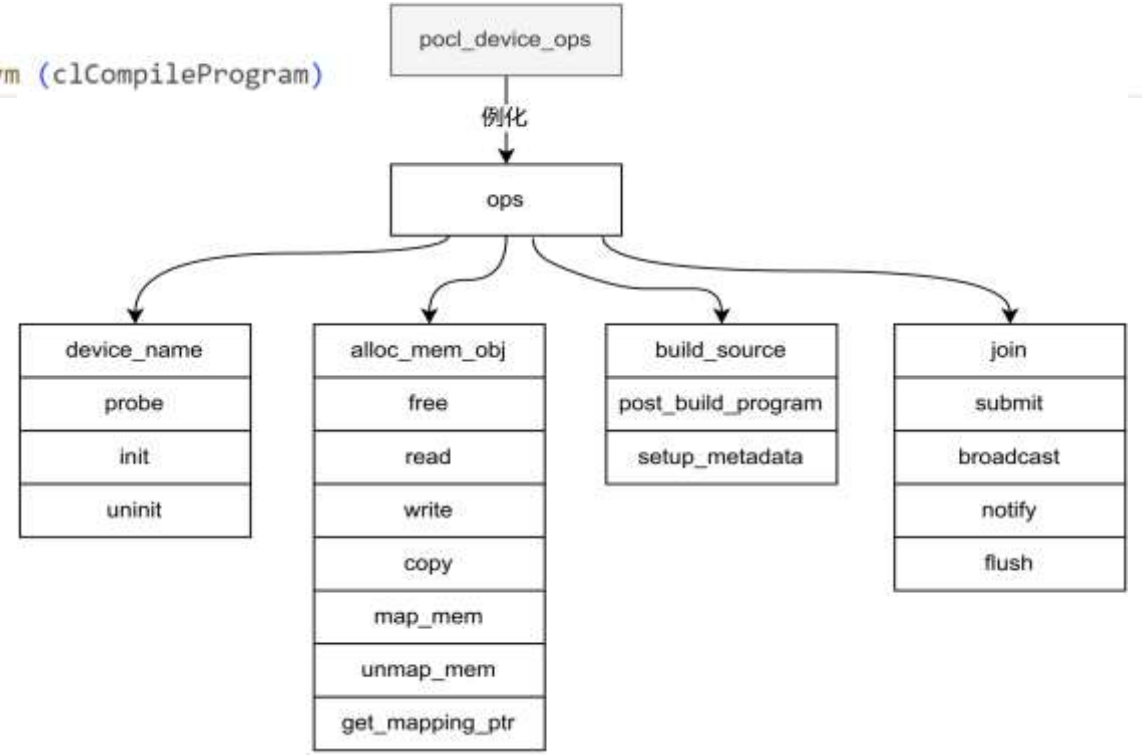
Runtime: OpenCL API实现

- 程序通过ocl-icd调用API实现
- 生成动态库libpocl.so

操作	API
获取平台中的计算设备信息	clGetDeviceIDs
创建OpenCL上下文	clCreateContext...*
创建设备和命令队列	clCreateCommandQueue
创建和构建程序对象	clCreatePrograemWith...
创建OpenCL内核	clCreateKernel
创建OpenCL缓冲区	clCreateBuffer
设置内核函数参数	clSetKernelArg
执行内核	clEnqueueNDRangeKernel

OpenCL关键API

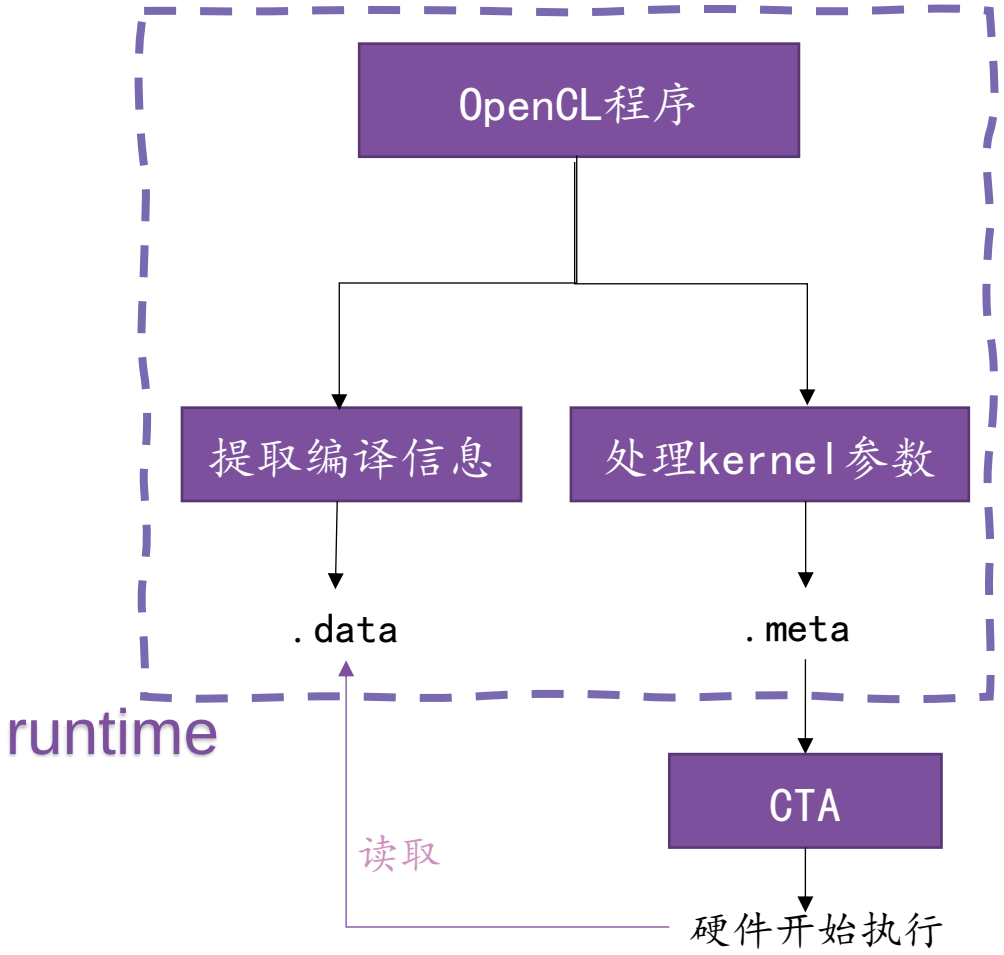
```
POname (clCompileProgram) (...)
CL_API_SUFFIX__VERSION_1_2
{

...
// callback to device's build function
device->ops->build_source(program, device_i, num_input_headers, input_headers,
                          header_include_names, (create_library ? 0 : link_program));
// finish callback
...
}
POsym (clCompileProgram)
```



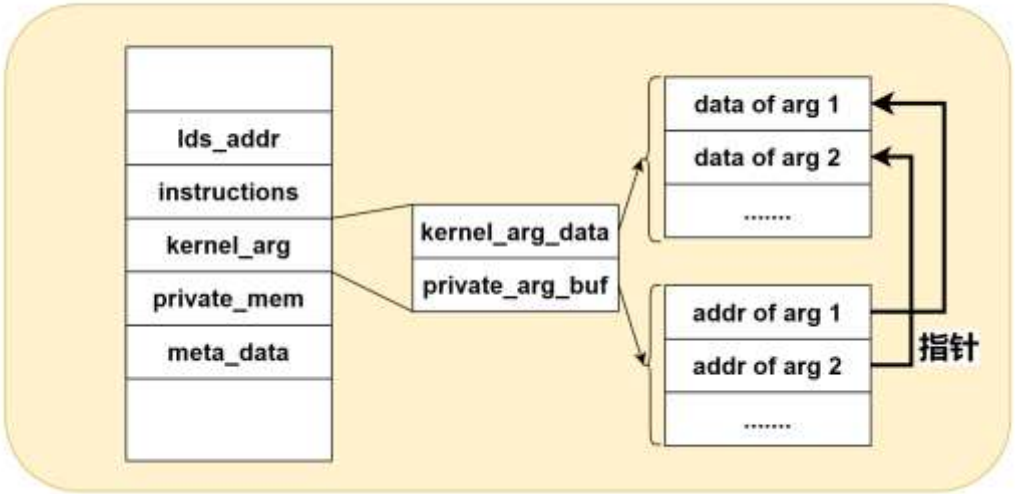
运行时环境需要实现的关键回调函数

Runtime: metadata实现

- 程序执行时，硬件需要一些动态的执行信息，由Runtime生成并传递。



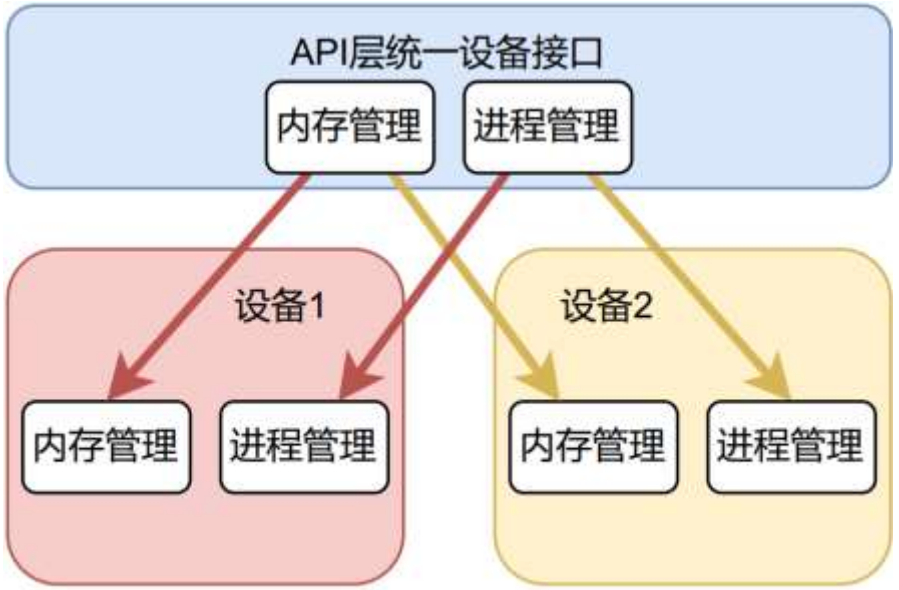
- 指针数组的形式索引kernel参数



metadata在内存中的布局

Driver：统一设备接口

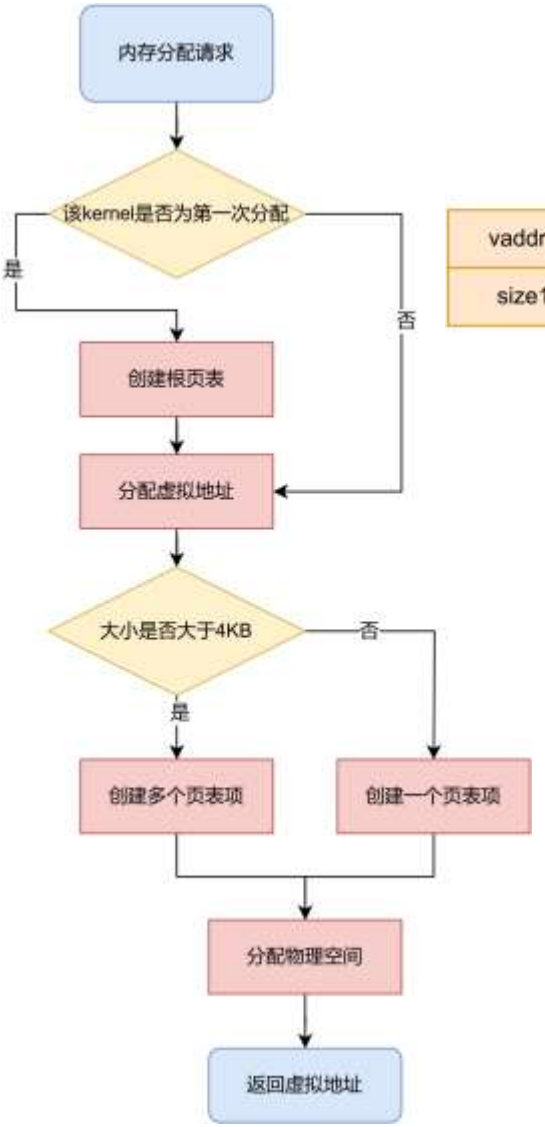
- 为上层软件提供统一接口，屏蔽下层差异
- 生成动态库libspike_driver.so



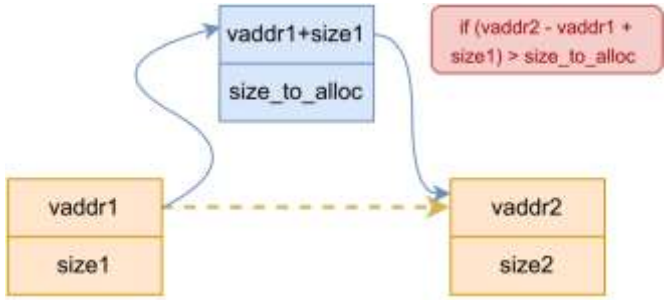
驱动程序整体架构

统一设备接口	描述
vt_dev_open	打开并连接一个 GPGPU 设备，调用该方法会初始化一个设备类，完成设备预留内存空间的初始化。
vt_dev_close	关闭一个 GPGPU 设备，调用该方法会等待设备执行完成目前所有任务，释放内存空间。
vt_root_mem_alloc	对于一个新的内核函数，创建一个根页表。
vt_buf_alloc	分配一块内存空间，返回一个设备端内存的虚拟地址
vt_buf_free	释放传入指针所指的内存空间，需要指定所属的内核函数
vt_copy_to_dev	将数据从主机端搬移到设备端
vt_copy_from_dev	将数据从设备端搬移到主机端
vt_start	启动一个内核函数，需要传入内核函数的 id 和 metadata，该方法会将内核函数拆分为多个线程块，然后调度到 GPGPU 设备上执行，任务调度模块需要在此实现

Driver：内存分配机制

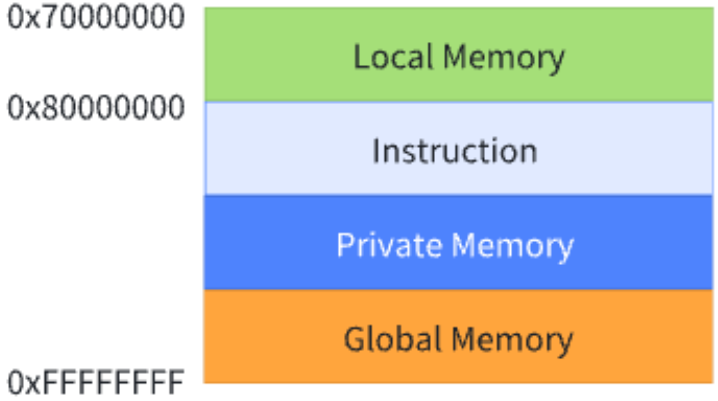


驱动程序的内存分配算法

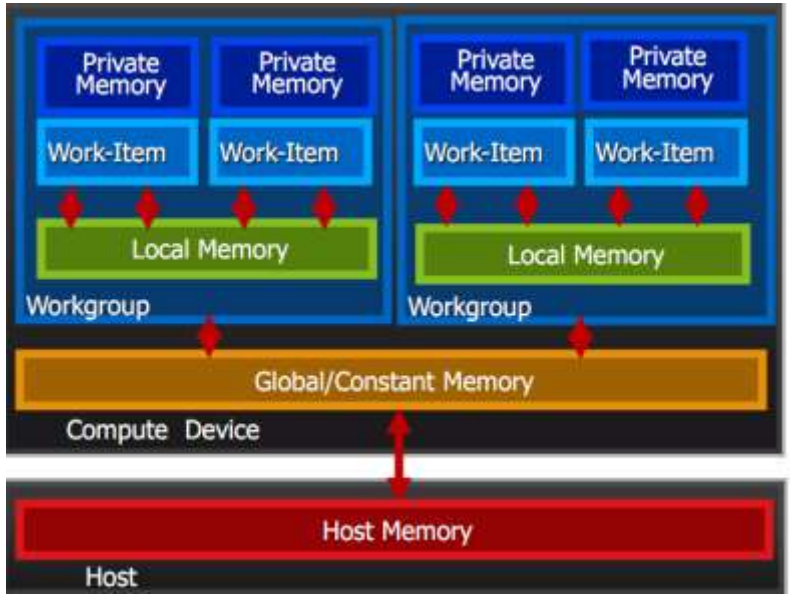


分配一个新的内存块

- 实现了将设备端内存映射到OpenCL内存模型的定义中
  - global memory
  - local memory
  - private memory
- 支持基于虚拟地址的内存空间分配



内存空间分配和映射



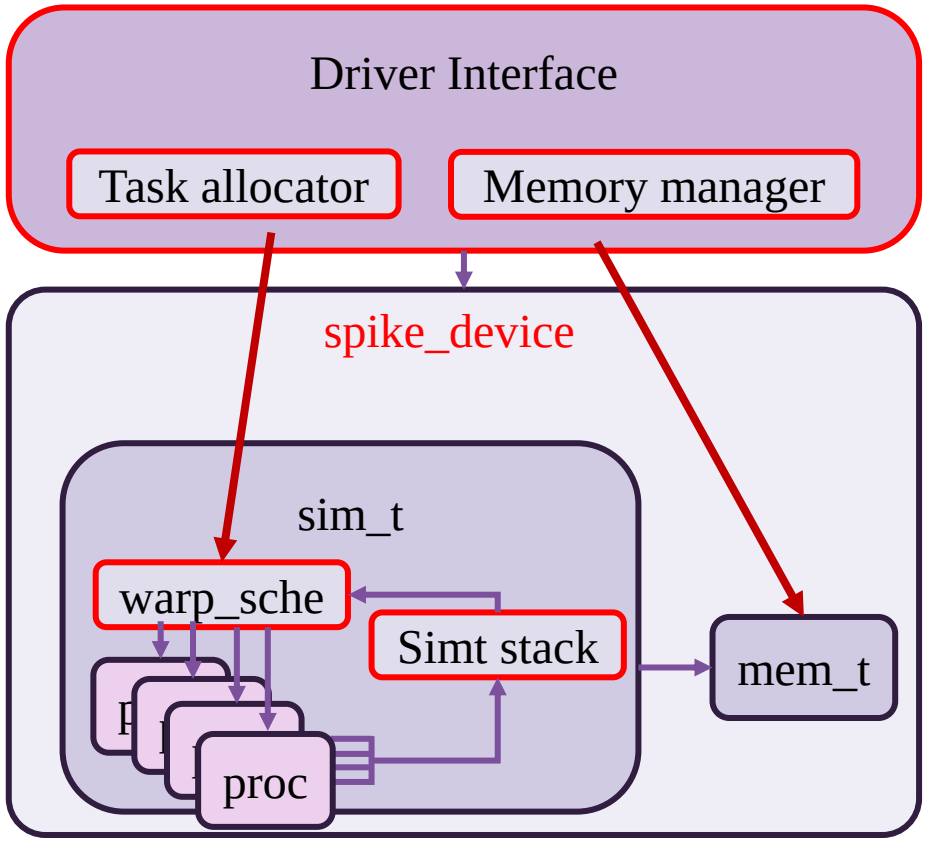
OpenCL内存模型

spike指令级仿真器

- 添加“乘影”的自定义指令

type	instruction name	usage
kernel response	endprog	endprg x0,x0,x0
synchronization	barrier, barriersub	barrier x0,x0,imm barriersub x0,x0,imm
branch control	vbeq, vbne, vblt vbge, vbltu, vbgeu	vbeq vs2, vs1, offset vbne vs2, vs1, offset
branch control	join, setrpc	join v0,v0,0 setrpc rd,rs1,offset
register index extension	regext,regexti	regext x0,x0,imm regexti x0,x0,imm
memory access	vlw.v, vlh(u).v, vlb(u).v vsw.v, vsh.v, vsb.v	vlw.v vd,offset(vs1) vsw.v vs2,offset(vs1)
memory access	vlw12.v, vlh(u)12.v, vlb(u)12.v vsw12.v, vsh12.v, vsb12.v	vlw12.v vd, offset(vs1) vsw12.v vs2,offset(vs1)
memory access	vadd12.vi	vadd12.vi vd, vs1, imm
tensor related	vfexp.v, vftta.v	vfexp vd,v2,v0.mask vftta.vv vd,v2,v1,v0.mask

“乘影”自定义指令



“乘影”指令级仿真器架构

- 实现与driver的接口
  - 支持内存系统及管理
- 实现GPGPU组件
  - simt-stack用于处理分支
  - GPGPU顶层模块
  - 线程束调度器

测试套

- OpenCL-CTS
  - 用于 OpenCL 兼容性测试
  - 总体通过率约80%

测试项目	用例数量	通过数	通过率
basic	115	107	93%
compiler	66	26	39%
api	105	96	91%
computeinfo	5	5	100%
commonfns	18	14	78%
relationals	17	17	100%
allocations	5	5	100%
buffers	93	82	88%
mem_host_flags	9	8	89%
events	28	24	86%
integer_ops	99	73	74%
workgroups	17	17	100%
geometrics	8	6	75%
vectors	9	8	89%
profiling	31	31	100%

OpenCL-CTS部分项目测试结果统计

- Rodinia benchmark
  - 涵盖不同领域，包括线性代数，图算法，神经网络训练等。

测试用例	来源	通过
vecadd	PoCL	√
matadd	PoCL	√
gaussian	Rodinia	√
nn	Rodinia	√
bfs	Rodinia	√
nw	Rodinia	√
kmeans	Rodinia	√
b+tree	Rodinia	√
backprop	Rodinia	√

执行效果

Kernel函数

```
__kernel void Fan1(__global float *m_dev,
                   __global float *a_dev,
                   __global float *b_dev,
                   const int size,
                   const int t) {
    int globalId = get_global_id(0);

    if (globalId < size-1-t) {
        *(m_dev + size * (globalId + t + 1)+t) =  \
        *(a_dev + size * (globalId + t + 1) + t) / *(a_dev +
size * t + t);
    }
}
```

支持自定指令的RISC-V汇编

```
800000ac <Fan1>:
addi        sp, sp, 16
sw          ra, -16(sp)
lw          t0, 12(a0)
sw          t0, -4(sp)
sw          a0, -12(sp)
lw          t0, 16(a0)
sw          t0, -8(sp)
vmv.v.x     v0, zero
jal         <get_global_id>
lw          s0, -8(sp)
lw          t2, -4(sp)
not         t0, s0
add         t0, t0, t2
vmv.v.x     v1, t0
```

```
auipc       t1, 0
setrpc      zero, t1, 108
vbge        v0, v1, <.LBB0_2>
lw          t1, -12(sp)
lw          t0, 4(t1)
lw          t1, 0(t1)
vmv.v.x     v1, s0
vmv.v.x     v2, t2
vmv.v.x     v3, t0
vadd.vv     v0, v0, v1
vadd.vi     v0, v0, 1
vmul.vv     v0, v0, v2
vsll.vi     v0, v0,
vadd.vv     v4, v3, v0
……
```

```
[2024-03-29 17:27:35.754596856] POCL: in fn void pocl_ventus_run(void *, _cl_command_node *) at line 803:
  *** INFO ***  |    VENTUS |  kernel metadata has been written to 0x90027000
to allocate at 0x90000000 with 8192 bytes
to copy to 0x90000000 with 4352 bytes
to allocate at 0x90002000 with 8192 bytes
notice that ventus hasn't support local buffer as argument yet.
to allocate at 0x90004000 with 4096 bytes
to copy to 0x90004000 with 16 bytes
to allocate at 0x90005000 with 131072 bytes
to allocate at 0x90025000 with 4096 bytes
to copy to 0x90025000 with 64 bytes
arg gpgpu is numw:8,numt:4,numwg:1,kernelx:1,kernely:1,kernelz:1,ldssize:0x1000,pdssize:0x10000000,pdsbas
vaddr mem scope is -m0x70000000:0x90026000
src file is object.riscv, run log is written to object.riscv.log
spike -l --log-commits -p8 --isa rv32gcv_zfh --pc=0x80000000 -m0x70000000:0x90026000 --varch vlen:128,ele
Log file object.riscv.log renamed successfully to kmeans_swap_0.log.
```

软件平台运行效果

工具链 — 搭建与使用

- 支持的系统镜像
  - Ubuntu22.04： ventus_ubuntu.Dockerfile
  - Centos7.9： ventus_centos.Dockerfile
- 发布版本
  - ventus-release-v2.1.0 （Ubuntu）
- Get started
  - 位置：【opengpgpu.org.cn】 - 【项目】
  - 构建方式：脚本一键生成
  - 使用方式
  - 硬件的运行方式
  - …

软件部分

1. 工具链的获取方式

- 方法一：参考构建说明，自行构建；
- 方法二：直接获取release版本，参考构建说明第2部分安装依赖工具；

注：建议采用ubuntu 22.04版本，否则可能会出现系统工具版本不兼容等问题；

2. 工具链的使用

- 单个warp中thread的数目num_thread，默认为32；在工具链使用之前，用户若有硬件需求
  adata文件中自动生成的nWarps的大小，建议num_thread尽量小）
- 可以结合构建说明的第3部分和用户的测试套使用说明，设置相应的环境变量，编译并运行测

注：这里运行用例，采用spike指令集仿真；

开源社区

https://opengpgpu.org.cn/

使命

促进 GPU 指令集、架构领域的开源开放协同创新

构建OpenGPGPU 的技术链、创新链和生态链

推动 GPU 产业健康快速发展以及其在社会经济各领域的广泛应用

开源社区

https://opengpgpu.org.cn/

- 代码仓

https://github.com/THU-DSP-LAB

https://www.gitlink.org.cn/THU-DSP-LAB

代码开源、issue、pr提交流程；

GitHub　　GitLink

- 资料

乘影架构文档手册：指令集架构及软硬件接口v202.pdf

乘影ISA介绍.pdf

乘影软件介绍.pdf

乘影硬件架构介绍.pdf

白皮书：承影GPGPU架构文档手册v2.02.pdf

欢迎大家加入开源社区！！！

BUG追踪

- **llvm-project** https://github.com/THU-DSP-LAB/llvm-project/issues
- **ventus-driver** https://github.com/THU-DSP-LAB/ventus-driver/issues
- **pocl** https://github.com/THU-DSP-LAB/pocl/issues
- **ventus-gpgpu-isa-simulator** https://github.com/THU-DSP-LAB/ventus-gpgpu-isa-simulator/issues
- **ventus-gpgpu** https://github.com/THU-DSP-LAB/ventus-gpgpu/issues
- **ventus-gpgpu-cpp-simulator** https://github.com/THU-DSP-LAB/ventus-gpgpu-cpp-simulator/issues

谢 谢！