Triton程序编写

嘉宾：杜玉博

Triton API语法

常见的Triton API语法

triton

`jit`	Decorator for JIT-compiling a function using the Triton compiler.
`autotune`	Decorator for auto-tuning a `triton.jit`'d function.
`heuristics`	Decorator for specifying how the values of certain meta-parameters may be computed.
`Config`	An object that represents a possible kernel configuration for the auto-tuner to try.

https://triton-lang.org/main/index.html

Triton API语法

triton.autotune

triton.autotune(*configs, key, prune_configs_by=None, reset_to_zero=None, restore_value=None, pre_hook=None, post_hook=None, warmup=25, rep=100, use_cuda_graph=False*)

Decorator for auto-tuning a `triton.jit`'d function.

```
@triton.autotune(configs=[
    triton.Config(kwargs={'BLOCK_SIZE': 128}, num_warps=4),
    triton.Config(kwargs={'BLOCK_SIZE': 1024}, num_warps=8),
  ],
  key=['x_size'] # the two above configs will be evaluated anytime
                 # the value of x_size changes
)
@triton.jit
def kernel(x_ptr, x_size, **META):
    BLOCK_SIZE = META['BLOCK_SIZE']
```

Note: When all the configurations are evaluated, the kernel will run multiple times. This means that whatever value the kernel updates will be updated multiple times. To avoid this undesired behavior, you can use the *reset_to_zero* argument, which resets the value of the provided tensor to *zero* before running any configuration.

Triton API语法

triton.Config

class **triton.Config**(*self*, *kwargs*, *num_warps=4*, *num_stages=2*, *num_ctas=1*, *maxnreg=None*, *pre_hook=None*)

An object that represents a possible kernel configuration for the auto-tuner to try.

Variables:

- **kwargs** – a dictionary of meta-parameters to pass to the kernel as keyword arguments.
- **num_warps** – the number of warps to use for the kernel when compiled for GPUs. For example, if *num_warps=8*, then each kernel instance will be automatically parallelized to cooperatively execute using *8 * 32 = 256* threads.
- **num_stages** – the number of stages that the compiler should use when software-pipelining loops. Mostly useful for matrix multiplication workloads on SM80+ GPUs.
- **num_ctas** – number of blocks in a block cluster. SM90+ only.
- **maxnreg** – maximum number of registers one thread can use. Corresponds to ptx .maxnreg directive. Not supported on all platforms.
- **pre_hook** – a function that will be called before the kernel is called. Parameters of this function are args.

Triton API语法

常见的Triton API语法

Math Ops

`abs`	Computes the element-wise absolute value of `x`.
`cdiv`	Computes the ceiling division of `x` by `div`
`ceil`	Computes the element-wise ceil of `x`.
`cos`	Computes the element-wise cosine of `x`.
`sin`	Computes the element-wise sine of `x`.
`softmax`	Computes the element-wise softmax of `x`.
`sqrt`	Computes the element-wise fast square root of `x`.

Triton API语法

常见的Triton API语法

Debug Ops

`static_print`	Print the values at compile time.
`static_assert`	Assert the condition at compile time.
`device_print`	Print the values at runtime from the device.
`device_assert`	Assert the condition at runtime from the device.

先进编译实验室
Advanced Compiler

Triton程序

向量加法示例

```
@triton.jit
def add_kernel(x_ptr, y_ptr, output_ptr,n_elements,  BLOCK_SIZE: tl.constexpr, ):
    pid = tl.program_id(axis=0)
    block_start = pid * BLOCK_SIZE
    offsets = block_start + tl.arange(0, BLOCK_SIZE)
    mask = offsets < n_elements
    x = tl.load(x_ptr + offsets, mask=mask)
    y = tl.load(y_ptr + offsets, mask=mask)
    output = x + y
    tl.store(output_ptr + offsets, output, mask=mask)
```

Triton程序

如何调用内核函数

```
def add(x: torch.Tensor, y: torch.Tensor):
    output = torch.empty_like(x)
    assert x.is_cuda and y.is_cuda and output.is_cuda
    n_elements = output.numel()
    grid = lambda meta: (triton.cdiv(n_elements, meta['BLOCK_SIZE']), )
    add_kernel[grid](x, y, output, n_elements, BLOCK_SIZE=1024)
    return output
```


先进编译实验室
Advanced Compiler

运行过程

编译运行过程：

```
# python 01-vector-add.py
```

运行结果分析：

```
(triton2.1.0) [duyb@Stream9 tutorials]$ python 01-vector-add.py
tensor([1.3713, 1.3076, 0.4940,  ..., 1.3374, 1.4960, 0.9115], device='cuda:0')
tensor([1.3713, 1.3076, 0.4940,  ..., 1.3374, 1.4960, 0.9115], device='cuda:0')
The maximum difference between torch and triton is 0.0
/public/home/duyb/anaconda3/envs/triton2.1.0/lib/python3.11/site-packages/triton/testing.py:304: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
vector-add-performance:
           size      Triton       Torch
0        4096.0   15.999999   15.999999
1        8192.0   24.000000   24.000000
2       16384.0   48.000000   48.000000
3       32768.0   90.352949   94.523077
4       65536.0  127.999995  127.999995
5      131072.0  192.000000  192.000000
6      262144.0  255.999991  250.137408
7      524288.0  279.272725  279.272725
8     1048576.0  307.200008  307.200008
9     2097152.0  319.168844  322.440352
10    4194304.0  327.714130  329.775424
11    8388608.0  332.108094  332.353736
12   16777216.0  334.367350  334.376254
13   33554432.0  335.508544  335.526441
14   67108864.0  335.938503  336.082067
15  134217728.0  336.225721  336.369531
```

AdvancedCompiler

Tel: 13839830713