Triton 踩坑

垃圾的文档

最近工作上需要优化一个自定义算子，用到了 OpenAI triton。但是查阅资料后，发现 triron 这个库的**文档**简直是**数一数二的垃圾**，就像那种只有数学公式，没有代码的论文一样。

个人认为，既然计算以 tensor 为单位的，就要把这个操作对应的输入、输出 tensor 形状写出来，然后给一个具体的例子，参考 PyTorch 的文档。而不是全都用自然语言来描述。

triton 是怎么做的呢?

举个例子：https://triton-lang.org/main/python-api/generated/triton.language.load.html#triton.language.load 这是 `tl.load` 的文档。里面提到了用 Block pointer 可以用 boundary check 和 padding option。

- 第一个问题就来了，boundary check 是做什么？是 out of bounds 就不加载了，获得一个比 Block shape 更小的 tensor?还是用某个值填满?还是运行时报错?还是什么别的操作?
- 接下来就是 padding option 的问题：这个 padding 指的是什么?然后还需要靠自己猜才知道，这个 padding 指的就是那些 out of bounds 的元素。

这里我是可以猜出来的，但**这本来应该是文档应该 explicit 写出来的，而不是用户来猜的**。

再比如，`tl.make_block_ptr` 和 `tl.arange`，这里的 block shape 的每个维度、和 arange 的元素个数，都必须是 2 的整数次幂。**这点居然没有在任何文档中体现出来，简直匪夷所思**。最后我是从代码报错中发现这个限制的，然后我去全网 google，在一篇非官方的、对 triton 进行讨论和 benchmark 的博客里面看到了这个限制。**总结来说，给 triton 写文档的人是真的对不起那帮做编译器的人**。

一些 API 和参数的 clarify

- `tl.load`
  - 如果你使用的是裸指针/很多裸指针组成的 tensor，请**同时设置 mask 和 other 参数**。
    - mask 为 True 代表这个位置要从 hbm 加载；False 代表使用 other 的值。
    - other 是一个浮点数。

    如果你是用的是 `tl.make_block_ptr` 创建的指针，请**同时设置所有维度的 boundary check，然后把 padding option 设置为 zero.**这样可以减少 bug 出现的概率，因为我们并不知道 boudary check 这个参数的 semantics 是怎么样的，尤其是在 make block pointer 设置了维度的顺序 order 之后。

- `tl.arange` 的元素个数、`tl.make_block_ptr` 的 block shape 都必须是 2 的整数次幂。或许任何 `triton` tensor 的每个维度都需要时 2 的整数次幂，但是我并没有进行测试。
- `tl.load` 和 `tl.store` 的 invalid memory access 都会导致 tensor 里面有的东西变成 nan。**没错，你没有看错，`tl.store` 也会**，不知道为什么，因为你访问了非法地址，原来合法的地址里面存的东西也变成了 nan！除非你保证你处理的维度都是 64 的倍数，否则请务必在从 HBM 读写数据的时候，把边界检查开起来。
  - 如果你是用的是裸指针，更要**小心设置 mask**。