Triton魔法入门(1)--Triton-Puzzles-Lite

前言

鉴于最近一系列有影响力的工作都用 `Triton` 进行了高性能的实现，决定趁寒假的空闲时间学习一下。

官网的 Tutorial 对于初学者而言稍微有些含混。在知乎遍览众多入门材料后，感觉 Sasha Rush 教授 的 Triton-Puzzles 比较有趣。在此基础上，@SiriusNEO 做了些改进（Triton-Puzzles-Lite），大幅减少依赖并且提高了题面的清晰度，加之更方便的调试，使其对初学者更加友好。

在这里记录下自己做这些 Puzzle 的解答和一些思路。

Puzzle 1: Constant Add

> Add a constant to a vector. Uses one program id axis. Block size `B0` is always the same as vector `x` with length `N0`.
>
> $z_i = 10 + x_i, \text{ for } i = 1, \ldots, N_0$

> ![[triton-puzzles-01.png]] 练手题，主要帮助掌握 `tl.load` 和 `tl.s

z_i = 10 + x_i, \text{ for } i = 1\ldots N_0

>![[triton-puzzles-02.png]] 在 `N0` 维度启动若干 threads，每个 threa

z_{j, i} = x_i + y_j,\text{ for } i = 1\ldots B_0,\ j = 1\ldots B_1

> ![[triton-puzzles-03.png]] 形式类似于向量外积，两个一维向量生成了一

z_{j, i} = x_i + y_j,\text{ for } i = 1\ldots N_0,\ j = 1\ldots N_1

> ![[triton-puzzles-04.png]] 加上 `pid` 和 `mask` 即可: ```python @

z_{j, i} = \text{relu}(x_i \times y_j),\text{ for } i = 1\ldots N_0,\ j = 1\ldots N_1

> ![[triton-puzzles-05.png]] 相比 Puzzle 4 外和变外积，又多了个 `relu

f(x, y) = \text{relu}(x_{j, i} \times y_j),\text{ for } i = 1\ldots N_0,\ j = 1\ldots N_1

dx_{j, i} = f_x'(x, y)_{j, i} \times dz_{j, i} $$ > > ![[triton-puzz