

CModel Of Bitlet

CModel的方式实现了Bitlet硬件加速算法的各个流程，使用了跟论文一样的6个weights作为示例进行演示

论文：Distilling Bit-level Sparsity Parallelism for General Purpose Deep Learning Acceleration

作者：中国科学院 路航 找不到解读

零基础入门SIMT Stack

SIMT Stack解决了当GPGPU执行遇到branch divergence时，通过Lane Mask屏蔽不需要被SIMD pipeline执行的func lane，串行地依次执行每一个分支的基本块代码，最终仍然可以给程序员提供每个线程独立执行的假象，同时根据SIMT条目中记录的基本块的IPDOM指示分化线程最早可能的reconvergence point，尽力提升程序的执行效率。

GPGPU模拟器自动打印出SIMT Stack的变化轨迹

零基础入门Likely-Convergence SIMT Stack：上一部分的进阶

基于GPGPU-SIM的MP\_IPDOM机制的实现：

读MP\_IPDOM论文

以上难以实现

要求复现论文内容，而GPGPU只是一个工具，不应该是复现内容，所以现在有两个方向：

1. Docker搭建GPGPU，然后找论文解读复现：找的论文不一定有bfs好理解 ×
2. 无奈之举，搭建GPGPU环境直接交付，需要找有没有这方面的论文
3. 世道变了：用并行bfs也是个不错的选择，不过需要两台电脑

GPGPU介绍：

也叫通用GPU，是图形处理单元，是用来辅助CPU的工具，帮助完成非图形相关程序的计算，为GPU负担一部分的非图形相关程序。能够用于物理计算、加密解密、科学计算和比特币加密货币生成。（可使用AIDA64查看GPGPU的详细信息）

GPGPU-sim介绍：

GPGPU-SIM模拟器的工作原理是让CUDA/OpenCL程序运行在模拟器上而不是GPU硬件。让我们只需要实现能让程序运行的部分。

这段代码模拟了一个简单的 GPU 执行流水线的行为，包括等待流操作、执行流操作、模拟时钟周期等。在每个循环迭代中，它检查 GPU 是否仍在执行任务，并在完成核函数时打印统计信息。

**理解SIMD和SIMT**

SIMT：单指令多线程

一个多核系统：每个核有自己的寄存器 ALU Cache

只有一个program couter寄存器和一个指令Cache和一个译码器

指令被广播到所有的SIMT核，有自己独立的栈核数据集

GPU←SIMT Core Cluster←SIMT Core←Thread Block←Thread

在CUDA中，Grid：并行执行线程块的集合，线程块：多个线程

线程块是SM的调度单元，线程束包含连续32个线程，是GPU调度的基本单元

SIMD：单指令多数据

小对比：两个相同元素个数的数组，将数组元素对应相加，多线程是分别分配一个线程，而多数据SIMD是分别分配一个ALU

**GPGPU-sim部署**

GPGPU-sim使用

编译CUDA源代码：将.cu文件复制到/configs/tested-cfgs目录中的某个文件里，对应某个型号的GPU

然后执行nvcc -cudart shared test.cu编译

用ldd查看可执行文件运行时库链接路径：ldd a.out

之后运行a.out文件，会有一堆输出信息，没有报错输出结果就能正常使用了