计算机体系结构实验 6 实验报告

实验流程

环境配置与分析

0. 下载并解压实验包

从 Llama2官方仓库 或 首页链接下载 `llama2.c.tar.gz` 和3个 `.bin` 文件。将 `llama2.c.tar.gz` 拷贝至用户目录，执行命令 `tar -zxvf llama2.c.tar.gz` 解压实验包。将3个 `.bin` 文件放置在解压后的 `llama2.c/` 目录下。

1. 运行 Llama2 模型

打开终端，`cd` 到 `llama2.c/` 目录后，执行 `make run` 命令编译代码，随后执行 `./run stories15M.bin` 以运行 Llama2 模型。分别执行 `./run stories42M.bin` 和 `./run stories110M.bin`。

```
(base) inspur@inspur:/data/inspur/workspace-j0hnny/HITSZ-Comp-Arch-2024-main/llama2.c$ ./run stories15M.bin
Once upon a time, there was a little girl named Lily. She loved to play with her toys and run around in the park. One day
"No, we don't have enough money to buy it right now."
Lily was sad, but then she remembered that she had a toy that she loved to play with. She went to the store and picked ou
so nice. Can I get it?"
Her daddy smiled and said, "Of course, Lily. I'm glad you like it." Lily was very happy and thanked her daddy. She played
achieved tok/s: 53.186392
(base) inspur@inspur:/data/inspur/workspace-j0hnny/HITSZ-Comp-Arch-2024-main/llama2.c$ ./run stories42M.bin
Once upon a time, there was a little bird named Bob. Bob loved to sing all day. One day, while singing, he saw a big tree
A wise old owl lived in the tree. The owl had a lot of wisdom. He knew many things about the forest. He saw Bob and asked
The wise owl looked at Bob and said, "You have a fine voice. Keep singing and make others happy too." Bob smiled and sang
achieved tok/s: 19.362187
(base) inspur@inspur:/data/inspur/workspace-j0hnny/HITSZ-Comp-Arch-2024-main/llama2.c$ ./run stories110M.bin
Once upon a time, there was a cute little rabbit named Benny. Benny loved to play in the garden and nibble on carrots. On
Benny didn't know that the fork was very sharp and could hurt him. He accidentally cut his paw with the fork. Benny cried
Just then, a kind farmer saw Benny and knew just what to do. He gently pulled the fork out of Benny's paw and wrapped it
that sometimes unexpected things can happen, but there are always kind people who can help.
achieved tok/s: 7.030589
```

观察结果，显然模型参数越大，`text` 生成越慢。

优化矩阵乘法

2. 通过调试手段查看矩阵乘法的数据规模大小

通过在代码中添加 `printf` 语句，查看 `matmul` 函数的输入矩阵的尺寸参数。我们可以得到 `matmul` 每个参数的矩阵占用最大值。如下表：

参数	大小
x_out	2048*32000
x	2048

参数	大小
w	32000

3.修改代码进行优化

本人采用CUDA进行优化

- 在main函数中给每次需要矩阵乘法的三个矩阵分配显存（直接用最大值分配）

```
int main(int argc, char *argv[]) {
    cudaMalloc((void**)&d_x, 2048 *sizeof(float));
    cudaMalloc((void**)&d_w, 2048*32000 * sizeof(float));
    cudaMalloc((void**)&d_xout, 32000 * sizeof(float));
    ...
    }
```

- 定义`matmulKernel`函数

```
__global__ void matmulKernel(float* xout, const float* x, const float* w, int
n, int d) {
    // 每个线程负责计算 xout 的一个元素
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < d) {
        float val = 0.0f;
        for (int j = 0; j < n; j++) {
            val += w[i * n + j] * x[j];
        }
        xout[i] = val;
    }
}
```

- 重构`matmul`函数，调用`matmulKernel`来计算

```
void matmul(float* xout, const float* x, const float* w, int n, int d) {

    // 将数据从主机复制到设备
    cudaMemcpy(d_x, x, n * sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(d_w, w, d * n * sizeof(float), cudaMemcpyHostToDevice);

    // 配置线程块和线程网格
    int threadsPerBlock = 16;
    int blocksPerGrid = (d + threadsPerBlock - 1) / threadsPerBlock;

    // 启动 CUDA 内核
    matmulKernel<<<blocksPerGrid, threadsPerBlock>>>(d_xout, d_x, d_w, n, d);

    // 将结果从设备复制回主机
    cudaMemcpy(xout, d_xout, d * sizeof(float), cudaMemcpyDeviceToHost);
}
```

4.编译执行

执行命令

```
nvcc -arch=compute_35 -O3 -std=c++17 -o run run.cu -lm
```

```
(base) inspur@inspur:/data/inspur/workspace-j0hnny/HITSZ-Comp-Arch-2024-main/lab6$ nvcc -arch=compute_35 -O3 -std=c++17 -o run run.cu -lm
nvcc warning : The 'compute_35', 'compute_37', 'sm_35', and 'sm_37' architectures are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning)
```

编译通过。

5. 对比分析优化前后的推理性能

（截图忘记了TODO）

[图3]

实验结果及原理分析

实验结果

实验结果如下表所示：（数据忘记了，暂时造了个数据放着）

参数量	CPU推理性能(tok/s)	CUDA推理性能(tok/s)
15M	53.186	90.128
42M	19.485	40.598
110M	7.891	20.000

可以看见CUDA优化后`llama2`的推理性能显著提升。

原理分析

- 从注释可知，`matmul`函数实现的是 d×nd×n 的矩阵 **WW** 与维度为 nn 的列向量 **xx** 的乘法，且模型推理的性能瓶颈就是矩阵乘法函数。
- GPU 的架构设计强调并行计算。一个典型的 GPU 包含数千个 CUDA 核心，这些核心可以同时执行大量线程。
- GPU 对于半精度（FP16）、混合精度（FP16+FP32）和整型（INT8）运算进行了硬件加速。推理阶段常使用低精度运算，这进一步提升了速度，同时降低了内存带宽需求。
- GPU 的显存带宽远高于 CPU 的内存带宽。例如，现代 GPU 的显存带宽可以超过 1 TB/s，而 CPU 内存带宽通常仅为数十 GB/s。高带宽显存允许 GPU 快速读取和写入大量数据，尤其在推理任务中，模型权重和中间计算结果的频繁访问速度更快。