MegEngine Meetup

一起聊聊有关底层优化工作的必备知识点

为什么要搞这么一期 meetup

- 正值校招，给大家支支招
- 很多人提到想做底层优化，那么底层优化到底需要些什么呢?

本次课程会涉及到以下内容

- Cache 结构
- 访存连续性
- 简单的分块优化
- SIMD / CPU Intrinsics
- 循环展开、OoO、指令级并行

What If: “你这也太基础了吧！”

您可能是 MegEngine 团队需要的人！

无论社招、校招还是实习，简历请发送至

xxr@megvii.com

好的面试题是如何考察能力的?

- 多层问题，层层递进
- 有一定开放性
- 不依赖于某个细节

Problem 1: 矩阵旋转 90 度

给定一个 $H \times W$ 的灰度图做为输入，请你将它顺时针旋转 90 度粘贴到另一个 $W \times H$ 的 buffer 中。（输入输出 buffer 空间已经开辟好）

```
void rotate(
    const uint8_t *input,
    uint8_t *output,
    size_t H,
    size_t W
)
```

Make it correct

$$i' = H - 1 - j$$

$$j' = i$$

```
for (size_t i=0; i < W; ++i) {
    for (size_t j=0; j < H; ++j) {
        // output[i][H-1-j] = input[j][i]
        output[(i * H)+H-1-j] = input[j * W + i];
    }
}
```

假如 H/W 比较大，以下四种实现的性能大概是怎么样的关系

- A: memcpy
- B: 顺时针旋转 90度
- C: 逆时针旋转 90度
- D: 顺时针旋转 180 度

假如 H=W，H 从 1 - 10000 时，拷贝速度的变化曲线如何？影响因素分别是什么？

如何写一个在任意 shape 情况下都足够快的方案呢?
如何定义“足够快”?

Problem 2： High dimemsion Reduce Sum

```
x = np.random.random((100, 100, 100)).astype(np.float32)

%timeit x.sum(axis=0)
%timeit x.sum(axis=1)
%timeit x.sum(axis=2)
```

如何编写能让上述三次代码的耗时基本接近？如何尽量提高速度?

以 axis=1 为例

```
x[dim1][dim2][dim3]
r[dim1][dim3]

for (size_t i = 0; I< dim1; ++i) {
  for (size_t k = 0; k < dim3; ++k) {
    for (size_t j = 0; j < dim2; ++j) {
      r[i][k] += x[i][j][k];
    }
  }
}
```

连续访存

```
for (size_t i = 0; I< dim1; ++i) {
  for (size_t j = 0; j < dim2; ++j) {
    for (size_t k = 0; k < dim3; ++k) {
      r[i][k] += x[i][j][k];
    }
  }
}
```

降低 miss rate

```
for (size_t i = 0; I< dim1; ++i) {
  for (size_t k = 0; k < dim3; k += 16) { // assert %16=0
    for (size_t j = 0; j < dim2; ++j) {
      r[i][k] += x[i][j][k];
    }
  }
}
```

循环展开

```
for (size_t i = 0; I< dim1; ++i) {
  for (size_t j = 0; j < dim2; ++j) {
    for (size_t k = 0; k < dim3; k += 4) { // assert %4=0
      r[i][k] += x[i][j][k];
      r[i][k+1] += x[i][j][k+1];
      r[i][k+2] += x[i][j][k+2];
      r[i][k+3] += x[i][j][k+3];
    }
  }
}
```

SIMD

```
for (size_t i = 0; I< dim1; ++i) {
  for (size_t j = 0; j < dim2; ++j) {
    for (size_t k = 0; k < dim3; k += 4) { // assert %4=0
      mmr = _mm_load_ps(&r[i][k]);
      mmx = _mm_load_ps(&x[i][j][k]);
      mmr = _mm_add_ps(mmr, mmx);
      _mm_save_ps(&r[i][k], mmr);
      //r[i][k] += x[i][j][k];
      //r[i][k+1] += x[i][j][k+1];
      //r[i][k+2] += x[i][j][k+2];
      //r[i][k+3] += x[i][j][k+3];
    }
  }
}
```

以上技能均可以组合使用

(记得想着内存要对齐

快问快答

性能优化的三种瓶颈

- computation bound
- throughput bound
- latency bound

1. 请分别举例
2. 如何将一份代码的瓶颈从 latency bound 转化成 computation / throughput bound?

这份代码为什么多线程执行时，加速比特别差?

```
int data[W * N];
int result[W];

void worker(int worker_id) {
   for (int i=0; i < N; ++i) {
     result[worker_id] += data[worker_id * N + i];
   }
}

int main() {
    // Start W worker
    // Wait W worker stop
    //print result
}
```

为什么推理的时候，GPU 比 CPU 需要开更大的 batch 才能吃满性能?

That's all

这是底层优化基础中的基础，实际应用中还需要更多的了解操作系统、CPU 等细节实现，来达到更高的性能。

想试试看自己的水平？来写个矩阵乘法吧~ 或者...来面个试？

【框架开发工程师（C++）】

职位描述

1. 负责旷视核心深度框架 MegEngine 的设计，演进，实现，维护和优化
2. 优化 MegEngine 在各个计算平台（CUDA / Arm / x86 等）上的性能
3. 持续探索改进深度学习框架的先进优化方法（例如图优化，自动代码生成，超低 bit 量化，稀疏化等）

技能要求

1. 1-3 年的性能优化经验（X86，CUDA，ARM，OpenCL 等）
2. 有良好的数据结构与算法功底，能够熟练使用 C++ 语言编写较复杂的算法
3. 对深度学习和深度学习框架（Caffe，Tensorflow，PyTorch等）有基本了解者优先
4. 有复杂系统设计经验者优先

感谢观看 & 期待你的加入

无论社招、校招还是实习，简历请发送至

xxr@megvii.com