Homework3

21307099 李英骏

2023.11.27

Problem 1

(a)

1. 基础吞吐量

   - 每个 *nondiverged SIMD instruction* 每 4 个周期产生 32 个结果
   - 每个 *SIMD processor* 包含 8 个通道, 说明每个通道每 4 个周期可以产生 $\frac{32}{8} = 4$ 个结果
   - 考虑到 80% 的线程活跃，每条指令实际产生的结果数为 $32 \times 0.8 = 25.6$Flops
   - 频率为 *2 GHz*，则每秒执行 $\frac{2\,\mathrm{G\ cycles}}{4} = 0.5\,\mathrm{G}$ cycles 组指令
   - 考虑到共 10 个处理器, 基础吞吐量为 $0.5\,\mathrm{G/s} \times 25.6\,\mathrm{G/s} \times 10 = 128\,\mathrm{GFLOPS/s}$

2. 考虑指令类型和发行率

   - 70% 的指令为单精度算术，且 *average SIMD instruction issue rate* 为 0.85, 则该内核在该 GPU 上的吞吐量为:

$$128\,\mathrm{GFLOPS/s} \times 70\% \times 0.85 = 76.16\,\mathrm{GFLOPS/s}$$

(b)

1. Increasing the number of single-precision lanes to 12

   - 若使用线性比例计算吞吐量的提升 (某些 GPU 可能优化了指令宽度和通道数量不对齐的情况), $Speedup = \frac{12}{8} = 1.5$
   - 但由于 32 不能被 12 整除, 即长度为 32 的向量被分为 12, 12, 8 三个周期处理, 而原先需要 4 个周期。则加速比为 $Speedup = \frac{4}{3} \approx 1.3333$

2. Increasing the number of SIMD processors to 15 (assume this change doesn't affect any other performance metrics and that the code scales to the additional processors)

   - 假设增加处理器数量不影响其他性能指标, 并且代码可以扩展到额外的处理器, 则 $Speedup = \frac{15}{10} = 1.5$

3. Adding a cache that will effectively reduce memory latency by 40instruction issue rate to 0.95%, which will increase

   - 缓存减少内存延迟 40%，提高 *instruction issue rate* 到 0.95, 则新的吞吐量为：
     $76.16\,\text{GFLOPS/s} \times \frac{0.95}{0.85} = 85.12\,\text{GFLOPS/s}$
   - $Speedup = \frac{0.95}{0.85} \approx 1.1176$

Problem 2

Assume that both the vector processor and GPU are performance bound by memory bandwidth. 因此认为向量处理器的性能足够高, 忽略向量处理器的执行时间。

向量计算机执行时间

- 标量部分的执行时间为 300 毫秒。
- 向量内核的执行时间：由于向量处理器的内存带宽为 30 GB/s，内核处理 400 MB 数据（300 MB 输入 + 100 MB 输出）。
- 向量内核执行时间 $= \frac{400\,\text{MB}}{30\,\text{GB/s}} = \frac{400\,\text{MB}}{30\times 2^{10}\,\text{MB/s}} \approx 13.0208\,\text{ms}$。
- 总执行时间 $= 300\,\text{ms} + 13.0208\,\text{ms} = 313.0208\,\text{ms}$

混合计算机执行时间

- 标量部分的执行时间为 300 毫秒，这是在主机处理器上执行的，时间是给定的，无需计算。
- 输入向量从主存载入 GPU 本地内存
  - 主机内存 -> GPU 本地内存
  - 300 MB
  - 使用 DMA 带宽（10 GB/s）：$\frac{300\,\text{MB}}{10\,\text{GB/s}} = 29.2969\,\text{ms}$
- 向量内核在 GPU 上执行：
  - GPU 内存 -> GPU 处理器 -> GPU 内存
  - 输入 300 MB + 输出 100 MB = 400 MB
  - 使用 GPU 内存带宽（150 GB/s）：$\frac{400\,\text{MB}}{150\,\text{GB/s}} = 2.6042\,\text{ms}$
- 输出向量从 GPU 本地内存传回主存：
  题中写的是 requires all *input vectors* to..., 我理解应该是相对的, 即运算完传输的是输出向量。
  - GPU 本地内存 -> 主机内存
  - 100 MB

  - 使用 DMA 带宽（10 GB/s）：$\frac{100\,\text{MB}}{10\,\text{GB/s}} = 9.7656\,\text{ms}$

- DMA 传输延迟：
  - 由于有两次传输（输入和输出），总延迟为 20 毫秒

- 总执行时间：

$$300\,\text{ms} + 29.2969\,\text{ms} + 2.6042\,\text{ms} + 9.7656\,\text{ms} + 20\,\text{ms} \approx 361.67\,\text{ms}$$

Problem 3

(a)

Assume that when processors are in use, the applications run times faster. 故修正 Amdahl 定律如下:

$$S(p) = \frac{1}{\sum_{i=1}^{p} \frac{F(i,p)}{i}}$$

(b)

由上述修正 Amdahl 定律:

$$\begin{aligned} S(8) &= \frac{1}{\left(F(1,8)\cdot 1 + F(2,8)\cdot\frac{1}{2} + F(4,8)\cdot\frac{1}{4} + F(6,8)\cdot\frac{1}{6} + F(8,8)\cdot\frac{1}{8}\right)} \\ &= \frac{1}{\left(\left(1 - \sum_{i=2}^{8} F(i,p)\right)\cdot 1 + 20\%\cdot\frac{1}{2} + 10\%\cdot\frac{1}{4} + 5\%\cdot\frac{1}{6} + 15\%\cdot\frac{1}{8}\right)} \\ &= \frac{1}{\left(50\%\cdot 1 + 20\%\cdot\frac{1}{2} + 10\%\cdot\frac{1}{4} + 5\%\cdot\frac{1}{6} + 15\%\cdot\frac{1}{8}\right)} \\ &\approx 1.5335 \end{aligned}$$

(c)

由上述修正 Amdahl 定律, 对于 32 处理器的情况:

$$\begin{aligned} S(32) &= \frac{1}{\left(F(1,32)\cdot 1 + F(2,32)\cdot\frac{1}{2} + F(4,32)\cdot\frac{1}{4} + F(6,32)\cdot\frac{1}{6} + F(8,32)\cdot\frac{1}{8} + F(16,32)\cdot\frac{1}{16}\right)} \\ &= \frac{1}{\left(\left(1 - \sum_{i=2}^{32} F(i,p)\right)\cdot 1 + 20\%\cdot\frac{1}{2} + 10\%\cdot\frac{1}{4} + 5\%\cdot\frac{1}{6} + 15\%\cdot\frac{1}{8} + 20\%\cdot\frac{1}{16}\right)} \\ &= \frac{1}{\left(30\%\cdot 1 + 20\%\cdot\frac{1}{2} + 10\%\cdot\frac{1}{4} + 5\%\cdot\frac{1}{6} + 15\%\cdot\frac{1}{8} + 20\%\cdot\frac{1}{16}\right)} \\ &\approx 2.1525 \end{aligned}$$

对于无穷个处理器的情况:

$$
\begin{aligned}
S(\infty) &= \frac{1}{\left(F(1,\infty)\cdot 1 + F(2,\infty)\cdot \frac{1}{2} + \cdots + F(\infty,\infty)\cdot \frac{1}{\infty}\right)} \\
&= \frac{1}{\left(F(1,\infty)\cdot 1 + F(2,\infty)\cdot \frac{1}{2} + F(4,\infty)\cdot \frac{1}{4} + F(6,\infty)\cdot \frac{1}{6} + F(8,\infty)\cdot \frac{1}{8} + F(16,\infty)\cdot \frac{1}{16} + F(128,\infty)\cdot \frac{1}{128}\right)} \\
&= \frac{1}{\left(20\%\cdot 1 + 20\%\cdot \frac{1}{2} + 10\%\cdot \frac{1}{4} + 5\%\cdot \frac{1}{6} + 15\%\cdot \frac{1}{8} + 20\%\cdot \frac{1}{16} + 10\%\cdot \frac{1}{128}\right)} \\
&\approx 2.7370
\end{aligned}
$$

Problem 4

(a)

```
for (i=0; i < 100; i++) {
    A[i] = B[2*i+4]; /* S1 */
    B[4*i+5] = A[i]; /* S2 */
}
```

$S2$ uses the value $A[i]$ computed by $S1$ in the same iteration

(b)

```
for (i=0; i < 100; i++) {
    A[i] = A[i] * B[i]; /* S1 */
    B[i] = A[i] + c;    /* S2 */
    A[i] = C[i] * c;    /* S3 */
    C[i] = D[i] * A[i]; /* S4 */
}
```

1. True Dependences (Read after Write)
   - S1 on A[i] followed by S2 on A[i].
   - S1 on A[i] followed by S4 on A[i].
   - S3 on A[i] followed by S4 on A[i].
2. Output Dependences (Write after Write)
   - S1 and S3 on A[i].
3. Antidependences (Write after Read)
   - S1 on B[i] followed by S2 on B[i].

- S1 on A[i] followed by S3 on A[i].
- S2 on A[i] followed by S3 on A[i].
- S3 on C[i] followed by S4 on C[i].

```
for(i=0;i<100;i++){
    A[i] = A[i] * B[i]; /*S1*/
    B1[i] = A[i] + c; /*S2*/
    A1[i] = C[i] * c; /*S3*/
    C1[i] = D[i] * A1[i]; /*S4*/
}
```

(c)

```
for (i=0; i < 100; i++) {
    A[i] = B[i] + C[i]; /* S1 */
    B[i+1] = D[i] + E[i]; /* S2 */
    C[i+1] = D[i] * E[i]; /* S3 */
}
```

S1 读取了 S2 在前一轮迭代的写入值. 在 iteration i 中 S2 计算了 B[i+1], 这在 iteration i+1 中被 S1 读取. S1 与 S3 中的 C[i] 和 C[i+1] 同理, 因此不是并行的

```
// 使用临时数组来避免写入依赖
int tempB[100], tempC[100];
tempB[0] = B[0];
tempC[0] = C[0];

for (i=0; i < 100; i++) {
    A[i] = B[i] + C[i]; // S1 不变
    if (i < 99) {
        tempB[i+1] = D[i] + E[i]; // S2 写入临时数组
        tempC[i+1] = D[i] * E[i]; // S3 写入临时数组
    }
}

// 循环结束后，将临时数组的内容复制回原数组
for (i=1; i < 100; i++) {
    B[i] = tempB[i];
    C[i] = tempC[i];
}
```

Problem 5

峰值单精度浮算吞吐量

假设所有内存延迟都被隐藏, 则峰值吞吐量仅考虑 FPU 的运算能力, 为

$$16 \times 16\,\text{Flops} \times 1.5\,\text{GHz} = 384\,\text{GFlops/s}$$

可持续性

若想要该吞吐量可持续, 则需要在每次浮点运算时输入 2 个单精度浮点数, 输出 1 个单精度浮点数, 由 IEEE 754 标准, 一个单精度浮点数为 32 位即 4 Bytes。所需的内存带宽为

$$384\,\text{GFlops/s} \times 3\,\text{floats/Flops} \times 4\,\text{Bytes/float} = 4608\,\text{GB/S} \gg 100\,\text{GB/S}$$

因此不可持续。