

Homework1

习题 1.1

为求全局总和例子中的 `my_first_i` 和 `my_last_i` 推导一个公式。需要注意的是：在循环中，应该给各个核分配数目大致相同的计算元素。（提示：先考虑 n 能被 p 整除的情况）。

因为 n 能整除 p ，所以可设，除最后一个核心外的其余核心区间长度

$x = n / p$ ，设 i 代表每个核心。

`My_first_i` = $x * (i - 1)$

`My_last_i` = $x * i$

PS: 若 n 不能整除 p ，则 $x = \lfloor n/p \rfloor$

习题 1.6

在下列情况中，推导出公式求出 0 号核执行接受与加法操作的次数。

- 最初的求全局总和的伪代码。
- 树形结构求全局总和。

制作一张表来比较这两种算法在总核数是 2、4、8、……、1024 时，0 号核执行的接收与加法操作的次数。

假设共有 N 个核

最初求和 $s = N - 1$

树形求和 $s = \lceil \log N \rceil$

核数	2	4	8	16	32	64	128	256	512	1024
最初求和	1	3	7	15	31	63	127	255	511	1023
树形求和	1	2	3	4	5	6	7	8	9	10

习题 2.2

请解释在 CPU 硬件里实现的一个队列，怎么使用可以提高写直达高速缓存（write-through cache）的性能。

因为队列是 FIFO，所以可只将队首和队尾元素放进缓存区，仅在插入删除时更新缓存区，避免频繁操作主存区。

习题 2.3

回顾之前一个从缓存读取二维数组的示例。请问一个更大矩阵和一个更大的缓存是如何影响两对嵌套循环的性能的？如果 $MAX = 8$ ，缓存可以存储 4 个缓存行，情况又会是怎样的？在第一对嵌套循环中对 A 的读操作，会导致发生多少次失效？第二对嵌套循环中的失效次数又是多少？

一个更大的矩阵会使两个嵌套循环缺失次数增多，一个更大的缓存会使第一个嵌套循环缺失次数减少，第二个嵌套循环缓存到一定程度才会缺失次数减少。

如果 $MAX = 8$ ，缓存可以存储 4 个缓存行

第一个循环一行缺失两次，共失效 $2 * 8 = 16$ 次

第二个循环一列缺失八次，共失效 $8 * 8 = 64$ 次

习题 2.16

- a. 假定一个串行程序的运行时间为 $T_{\text{串行}} = n^2$ ，运行时间的单位为毫秒。并行程序的运行时间为 $T_{\text{并行}} = n^2/p + \log_2(p)$ 。对于 n 和 p 的不同值，请写出一个程序并找出这个程序的加速比和效率。在 $n = 10、20、40、...、320$ 和

$p = 1, 2, 4, \dots, 128$ 等不同情况下运行该程序。当 p 增加、 n 保持恒定时，加速比和效率的情况分别如何？当 p 保持恒定而 n 增加呢？

b. 假设 $T_{\text{并行}} = T_{\text{串行}}/p + T_{\text{开销}}$ ，我们固定 p 的大小，并增加问题的规模。

- 请解释如果 $T_{\text{开销}}$ 比 $T_{\text{串行}}$ 增长得慢，随着问题规模的增加，并行效率也将增加。
- 请解释如果 $T_{\text{开销}}$ 比 $T_{\text{串行}}$ 增长得快，随着问题规模的增加，并行效率将降低。

a)

```
#define N 320
void Matrix(vector<vector<int>> & a, int x) {
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            a[i][j] += x;
}
```

p 增加， n 保持恒定，加速比先增后降，效率降低。

p 保持恒定， n 增加，加速比增加，效率增加。

b)

$$E = \frac{T_{\text{串行}}}{p T_{\text{并行}}} = \frac{T_{\text{串行}}}{p (T_{\text{串行}}/p + T_{\text{开销}})} = \frac{1}{1 + p \frac{T_{\text{开销}}}{T_{\text{串行}}}}$$

如果 T （开销）比 T （串行）增长得慢，随着问题规模的增加， pT （开销）/ T （串行）越来越小，并行效率增加

如果 T （开销）比 T （串行）增长得快，随着问题规模的增加， pT （开销）/ T （串行）越来越大，并行效率降低

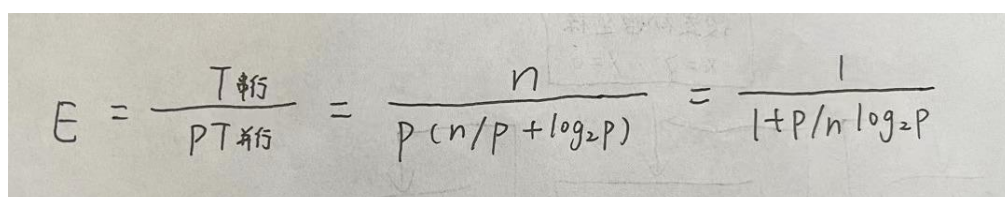
习题 2.17

如果一个并程序所获得的加速比可以超过 p （进程或线程的个数），则我们有时称该并程序拥有超线性加速比（superlinear speedup）。然而，许多作者并不讲能够克服“资源限制”的程序视为是拥有超线性加速比。例如，当一个程序运行在一个单处理器系统上时，它必须使用二级存储，当它运行在一个大的分布式内存系统上时，它可以将所有数据都放置在主存上。请给出另外一个例子，说明程序是如何克服资源限制，并获得大于 p 的加速比的。

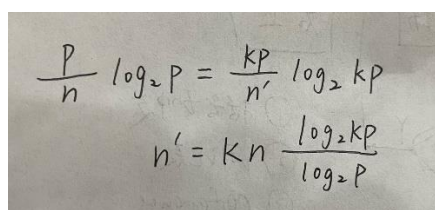
某串行程序设计得很差，比如读矩阵没有用到空间局部性原理，读取速率很慢，资源不断重复加载。但设计成并程序时，多核直接各读取一个元素，进程/线程可共享，获得超线性加速比。

习题 2.19

假定 $T_{\text{串行}} = n$, $T_{\text{并行}} = n/p + \log_2(p)$ ，时间单位为毫秒。如果以倍率 k 增加 p ，那么为了保持效率值得恒定，需要如何增加 n ？请给出公式。如果我们将进程数从 8 加倍到 16，则 n 的增加又是多少？该并程序是可扩展的吗？


$$E = \frac{T_{\text{串行}}}{p T_{\text{并行}}} = \frac{n}{p(n/p + \log_2 p)} = \frac{1}{1 + p/n \log_2 p}$$

因为效率恒定，所以 $p/n(\log(p))$ 恒定。


$$\frac{p}{n} \log_2 p = \frac{kp}{n'} \log_2 kp$$
$$n' = kn \frac{\log_2 kp}{\log_2 p}$$

上图是增加 n 的公式，如果进程数从 8 加到 16， n 增大了 $2 \cdot \log 16 / \log 8 = 8/3$ 倍。该并程序是可扩展的，因为对于任意一个 n ，都可以找到一个对应 p 求解。

习题 2.20

一个可以获得线性加速比的程序是强可扩展的吗？请解释。

是的。获得线性加速比时， $S=p$ ，加速比与问题规模没有关系，问题规模就可以认为是一个常数，且这个常数不随 p 的变化而变化。增加进程/线程数，不增加问题的规模，可以维持固定的并行效率，所以是强可扩展的。