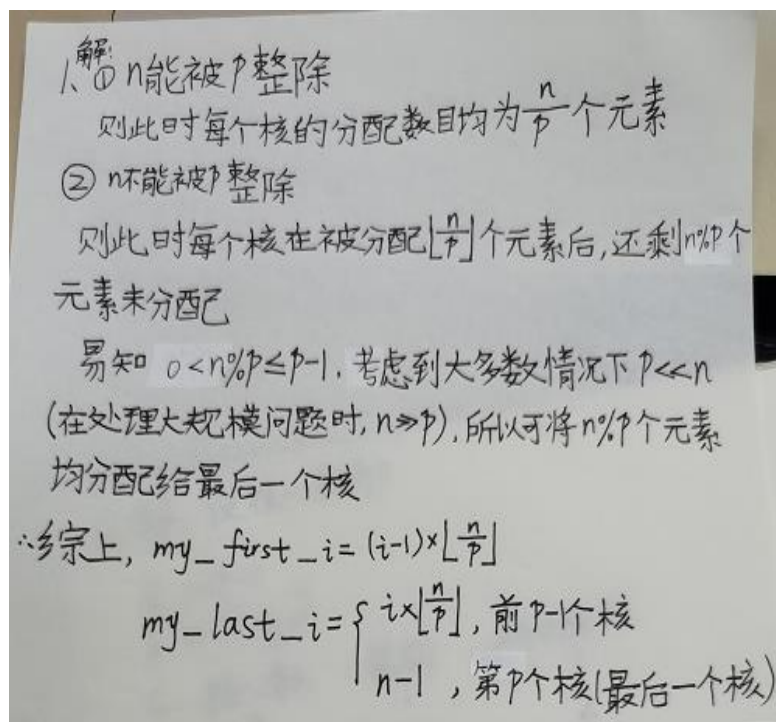


超级计算机原理与操作 Homework 1

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）。

解：



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。

解：见下页

2. 解: 设核心数为 P

a. $n = P - 1$
b. $n = \lceil \log_2 n \rceil$

c. 核心数	最初	树形结构
2	1	1
4	3	2
8	7	3
16	15	4
32	31	5
64	63	6
128	127	7
256	255	8
512	511	9
1024	1023	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。

解：在CPU硬件中，我们可以实现一个队列。当CPU需要写内存时，我们将其插入到队列尾部，但此时CPU不需要继续等待，而是继续执行接下来的指令。同时只要当前队列不为空，该队列就会一直以较慢的访存速度从队头取出元素并写入内存。这样大大减少了CPU因访存时间较慢而产生的等待时间，从而提高了性能。

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？

解：代码如下图所示

```

double A[MAX][MAX], x[MAX], y[MAX];
...
/* Initialize A and x, assign y = 0 */
...
/* First pair of loops */
for (i = 0; i < MAX; i++)
    for (j = 0; j < MAX; j++)
        y[i] += A[i][j]*x[j];
...
/* Assign y = 0 */
...
/* Second pair of loops */
for (j = 0; j < MAX; j++)
    for (i = 0; i < MAX; i++)
        y[i] += A[i][j]*x[j];

```

设cache-line的大小为 n 个元素，且采用直接映射。

对于第一对嵌套循环：

开始时cache为空，则在访问 $A[0][0]$ 时发生 1 次缺失，此时将 $A[0][0]$ 及其之后的 n 个元素读入cache中，则之后的 $n-1$ 次访问都能够命中。再接下来又会发生 1 次缺失，再次读取 n 个元素进入cache中。当cache满时，新的 n 个元素会替换cache某一行中之前的 n 个元素。

所以综上，第一对嵌套循环中，失效次数 = $(MAX * MAX) / n$ ，矩阵越大，失效次数越多；同时失效次数只与矩阵大小和cache行数有关，与缓存大小无关。

对于第二对嵌套循环：以 $j = 0$ 的第一层循环为例，有 MAX 组分别以 $A[0][0]$, $A[1][0]$, $A[2][0]$, ..., $A[MAX][0]$ 开头的 n 个元素会加入到cache中（cache每一行的第一次访问均会发生缺失）。所以若cache不够大，则之后的 n 个元素会替换之前的 n 个元素，这会导致在 $j = 1$ 的循环中访问到前面的元素时再次发生缺失，该元素又需要重新访存才能加载到cache中。

所以综上，第二对嵌套循环中，矩阵越大，失效次数越多；缓存越大，失效次数越少（发生cache替换的次数降低）。

当 $MAX = 8$ ，缓存可以存储 4 个缓存行时：

对于第一对嵌套循环： $x = 8 * 8 / 4 = 16$ ；

对于第二对嵌套循环： $x = 8 * 8 = 64$ ，即每次访问均缺失。

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 $n = 10, 20, 40, \dots, 320$ 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。

解：a. 对应程序如下图所示

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int n,p;
    double T1,T2,S,E;
    for(n=10;n<=320;n=n*10){
        for(p=1;p<=128;p=p*2){
            T1=n*n;
            T2=n*n/p+log(p);
            S=T1/T2;
            E=S/p;
            cout<<S<<" "<<E<<endl;
        }
        cout << endl;
    }
    system("pause");
}
```

分析运行结果可知：

p 增加， n 保持恒定时。加速比不断增加，效率不断降低。

p 保持恒定， n 增加时。加速比不断增加逼近于 p ，效率不断增加逼近于 1。

b.

$$b. E = \frac{T_{\text{串}}}{p \times T_{\text{并}}} = \frac{T_{\text{串}}}{T_{\text{串}} + p T_{\text{开销}}} = \frac{1}{1 + p \times \frac{T_{\text{开销}}}{T_{\text{串}}}}$$

① 若 $T_{\text{开销}}$ 比 $T_{\text{串}}$ 增长得慢，则 $\frac{T_{\text{开销}}}{T_{\text{串}}}$ 不断减小， $\frac{1}{1 + p \times \frac{T_{\text{开销}}}{T_{\text{串}}}}$ 不断增大，即 E 不断增加

② 若 $T_{\text{开销}}$ 比 $T_{\text{串}}$ 增长得快，则 $\frac{T_{\text{开销}}}{T_{\text{串}}}$ 不断增加， $\frac{1}{1 + p \times \frac{T_{\text{开销}}}{T_{\text{串}}}}$ 不断减小，即 E 不断下降

习题 2.17

如果一个并行程序所获得的加速比可以超过 p （进程或线程的个数），则我们有时称该并行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。

解：一个串行程序在运行时可能需要在不同阶段多次加载同一份资源。这时并行程序可以只加载一次并通过共享内存等方式来在不同线程之间共享该资源，从而减少了加载资源的时间，并借此操作来实现超线性加速比。

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？该并行程序是可扩展的吗？

解：

解： $E = \frac{T_{\text{串行}}}{PT_{\text{并行}}} = \frac{n}{n + p \log_2 p}$

当 p 变为 kp 时，
设 n 变为 mn

$$\therefore E' = \frac{mn}{mn + kp \log_2(kp)}$$

又 $E = E'$

$$\therefore m = \frac{k \log_2(kp)}{\log_2 p}$$

\therefore 以倍率 $\frac{k \log_2(kp)}{\log_2 p}$ 增加 n 即可

由题， $k = \frac{16}{8} = 2$ ， $p = 8$

$$\therefore m = \frac{2 \times 4}{3} = \frac{8}{3}$$

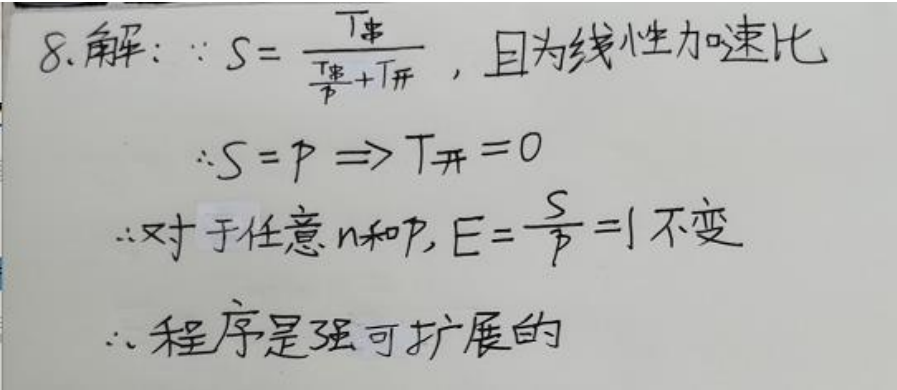
所以 n 增加至 $\frac{8}{3}n$ 即可

\therefore 易证该并行程序是可扩展的

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。

解：



8. 解：∵ $S = \frac{T_{\text{串}}}{\frac{T_{\text{串}}}{P} + T_{\text{开}}}$ ，且为线性加速比
∴ $S = P \Rightarrow T_{\text{开}} = 0$
∴ 对于任意 n 和 P ， $E = \frac{S}{P} = 1$ 不变
∴ 程序是强可扩展的