13.2　实现矩阵相乘的并行计算

矩阵A, B 均为N*N的方阵，试计算矩阵C=AB；
使用P个进程并行计算（N可以被P整除）；　$C_{ij}=\sum_{k=1}^{N}A_{ik}B_{kj}$
矩阵A，B及C均采用分布式存储；
A, C按行分割，　B按列分割存储（见本稿 47页）。
要求编写计算C矩阵的MPI程序，并进行计算。

实际计算时，矩阵A, B请采用如下值，　N设为10000

$$A_{ij}=e^{y_j}\sin 3x_i;B_{ij}=(x_i+\cos 4x_i)(1+y_j);x_i=(i-1)/(N-1);y_j=(j-1)/(N-1)$$

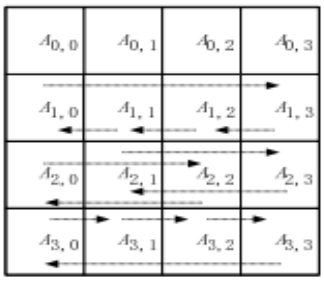
计算出C矩阵后，请计算 $S=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}c_{ij}$ ，并由根节点打印出来。

将S值与串行程序的结果进行对比，校验程序的正确性；
使用1,2,4,10个进程进行计算，并利用MPI_Wtime( )函数计算程序的运行时间；考核加速比及计算效率。

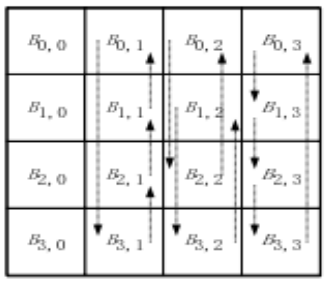
要求：　1）提交计算程序；　2）使用1，2，4，10个进程计算，提交计算结果（S值及计算时间）、计算效率及加速比。

算法说明：

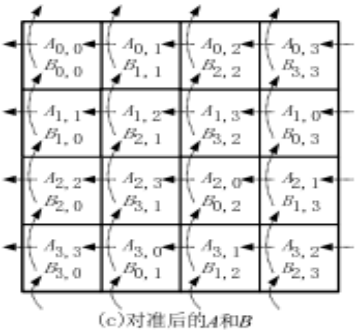
由于这道题目在 HPC 课上也有记录，故采用行列分块算法，每个进程内含有 A 矩阵一小块的行，含有 B 矩阵一小块的列。具体过程如下所示。



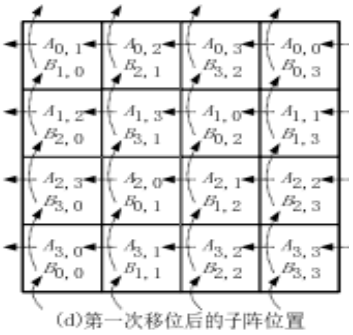
(a)*A*阵起始对准



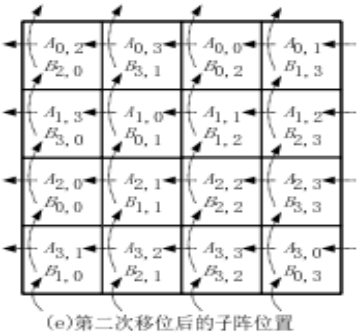
(b)*B*阵起始对准



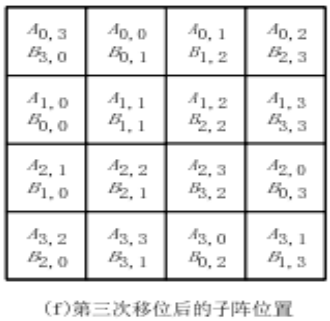
(c)对准后的*A*和*B*



(d)第一次移位后的子阵位置



(e)第二次移位后的子阵位置



(f)第三次移位后的子阵位置

求解思路：

1、首先规划笛卡尔网格进程，对每个进程进行赋值

2、通过 MPI 打包函数将单核的赋值数据与多核的赋值数据进行对比，确认赋值正确

3、开始计时，并且通过 Canno 算法对矩阵乘进行求解

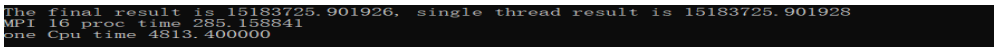
4、根进程求解全局矩阵得到时间

由于该算法需要$\sqrt{P}$个进程数目，故在 10000*10000 计算时采用 1，4，16，25 个处理器对其进行测试，结果如下，其中

加速比=单个进程执行的时间/N 个进程执行的时间；并行效率=加速比/N，单核采用的算法为$O(n^3)$的矩阵乘，未做优化。

	1	4	16	25	64
S	15183725.9	15183725.9	15183725.9	15183725.9	15183725.9
计算时间(s)	4813.4	772	285	287	269
计算效率		1.55	1.03	0.67	0.27
加速比		6.2	16.88	16.77	17.89

16 核计算实例：



由于电脑只有 8 个核心，多了也是虚拟出来的进程的，多余八个后会分摊掉原有的八个核的性能，所以后来加速比不理想，如要测试应当由确切的实际数量核心的测试平台。除此之外，效果相比商业计算软件不佳的原因是在计算中全局矩阵传递时，采用的是 MPI_Sendrecv_replace 函数，该函数无需各个进程缓冲区，直接替换数据，在这个过程中需要大量时间同步，除此之外可能算法优化还有数学处理上，商业软件来的更好。

附 Ryzen 7 5800H

CPU 核心数量	8
基准时钟频率	3.2GHz
最大加速时钟频率	4.4GHz
默认 TDP/TDP	45W

TLB\Reorder Buffer 大小暂未查询到