Dense Matrix Multiplication

代码片段是可以并行的。

拓展后的代码：

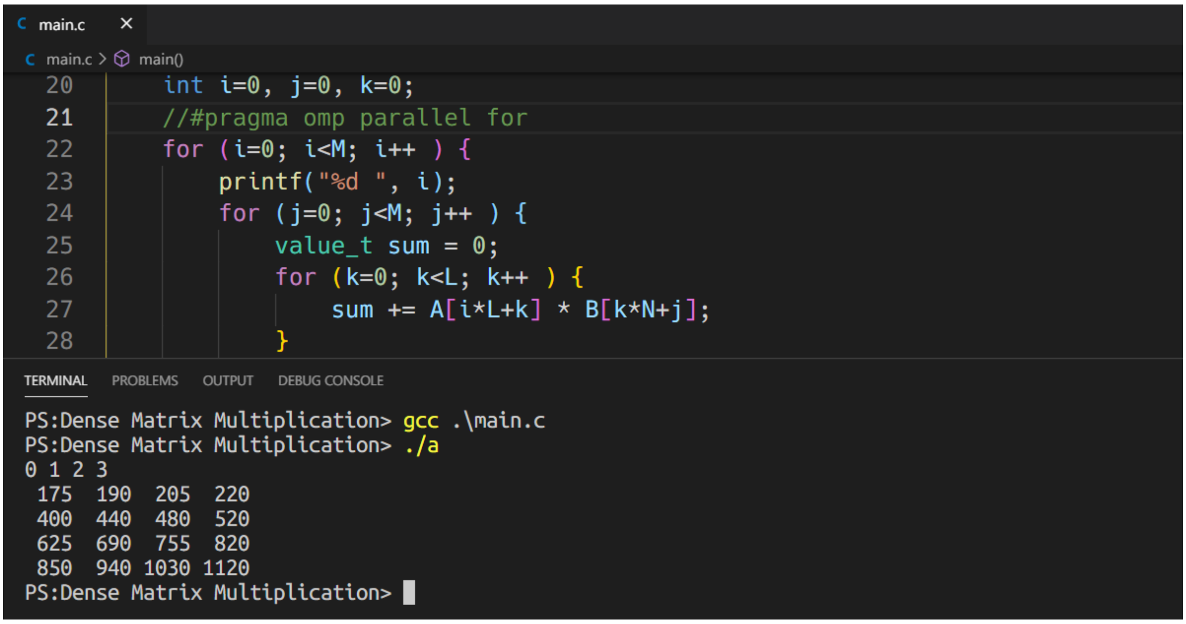
```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

typedef int value_t;

int main(){
    int M = 4;
    int L = 5;
    int N = 4;

    // value_t *A = (value_t*)malloc(sizeof(value_t) * M * L);
    // value_t *B = (value_t*)malloc(sizeof(value_t) * L * N);
    // value_t *C = (value_t*)malloc(sizeof(value_t) * M * N);

    value_t A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
    value_t B[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
    value_t *C = (value_t*)malloc(sizeof(value_t) * M * N);

    int i=0, j=0, k=0;
    #pragma omp parallel for
    for (i=0; i<M; i++ ) {
        printf("%d ", i);
        for (j=0; j<M; j++ ) {
            value_t sum = 0;
            for (k=0; k<L; k++ ) {
                sum += A[i*L+k] * B[k*N+j];
            }
            C[i*N+j] = sum;
        }
    }
    printf("\n");

    for(i=0; i<M; i++){
        for(j=0; j<N; j++){
            printf(j ? " %4d" : "%4d", C[i*M+j]);
        }
        printf("\n");
    }

    return 0;
}
```

关键代码：

```
1       #pragma omp parallel for
2       for (i=0; i<M; i++ ) {
3           printf("%d ", i);
4           for (j=0; j<M; j++ ) {
5               value_t sum = 0;
6               for (k=0; k<L; k++ ) {
7                   sum += A[i*L+k] * B[k*N+j];
8               }
9               C[i*N+j] = sum;
10          }
11      }
12      printf("\n");
```

其中第3行的printf语句是为了判断并行究竟有没有发生

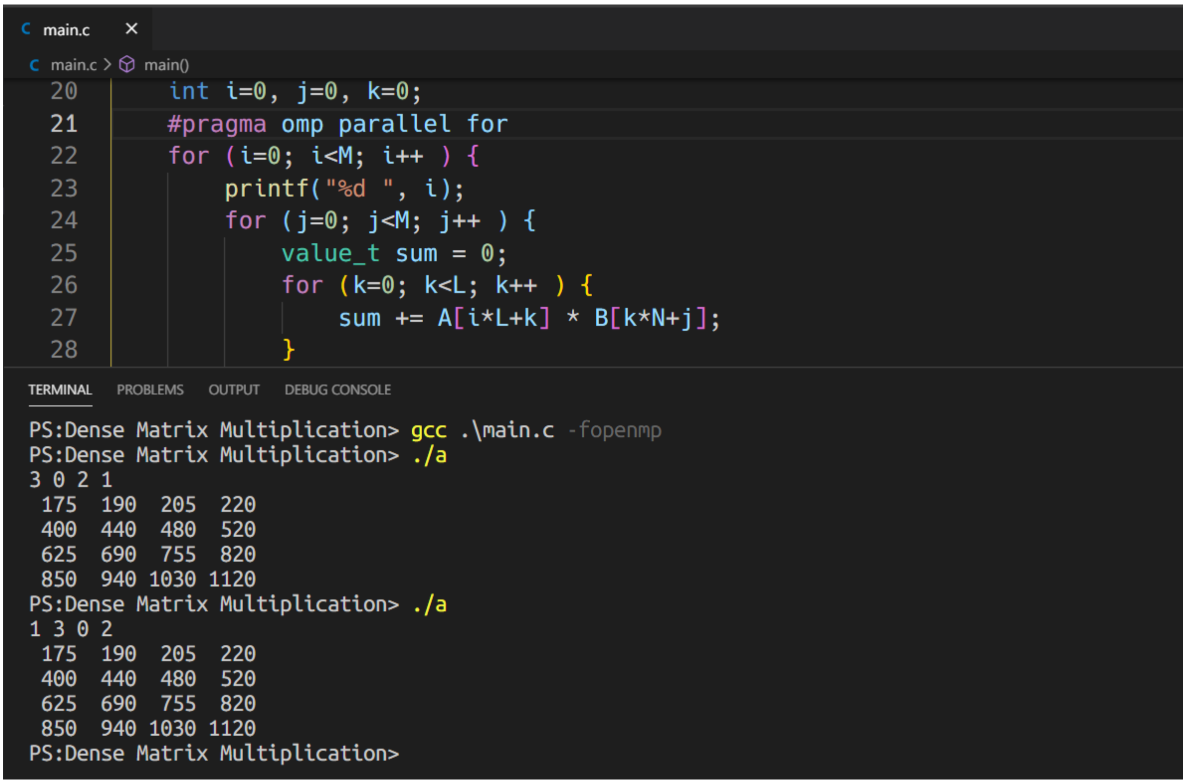
在注释掉第一行openmp标记的情况下编译运行:



```
    int i=0, j=0, k=0;
    //#pragma omp parallel for
    for (i=0; i<M; i++ ) {
        printf("%d ", i);
        for (j=0; j<M; j++ ) {
            value_t sum = 0;
            for (k=0; k<L; k++ ) {
                sum += A[i*L+k] * B[k*N+j];
            }
```

```
PS:Dense Matrix Multiplication> gcc .\main.c
PS:Dense Matrix Multiplication> ./a
0 1 2 3
 175  190  205  220
 400  440  480  520
 625  690  755  820
 850  940 1030 1120
PS:Dense Matrix Multiplication>
```

在添加openmp标记的情况下编译运行:



```
    int i=0, j=0, k=0;
    #pragma omp parallel for
    for (i=0; i<M; i++ ) {
        printf("%d ", i);
        for (j=0; j<M; j++ ) {
            value_t sum = 0;
            for (k=0; k<L; k++ ) {
                sum += A[i*L+k] * B[k*N+j];
            }
```

```
PS:Dense Matrix Multiplication> gcc .\main.c -fopenmp
PS:Dense Matrix Multiplication> ./a
3 0 2 1
 175  190  205  220
 400  440  480  520
 625  690  755  820
 850  940 1030 1120
PS:Dense Matrix Multiplication> ./a
1 3 0 2
 175  190  205  220
 400  440  480  520
 625  690  755  820
 850  940 1030 1120
PS:Dense Matrix Multiplication>
```

可以看到在未使用openmp时是顺序计算for循环的，而在使用了openmp后for循环计算顺序变得不确定，但最后得结果都是正确的。

可行的并行策略：

对于C矩阵的每个位置，计算该位置上的值，这些值之间并无依赖，因此可以并行计算。