به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

برنامه‌نویسی چندهسته‌ای

گزارش کار آزمایش ۲

سید محمد حجازی حسینی

۹۷۳۳۰۲۰

بهار ۱۴۰۱

نتایج حالت سریال)

اندازه ماتریس (سریال)			
512 x 512	2048 x 2048	8192 x 8192	10000 x 10000
0.001000	0.015600	0.253900	0.374100

نتایج روش اول)

تعداد نخ	اندازه ماتریس (موازی)			
	512 x 512	2048 x 2048	8192 x 8192	10000 x 10000
۱	0.001800	0.025900	0.402100	0.600300
۲	0.001200	0.014200	0.226200	0.329300
۴	0.000900	0.010300	0.129800	0.193500
۸	0.000800	0.008400	0.113000	0.165400

تعداد نخ	تسریع			
	512 x 512	2048 x 2048	8192 x 8192	10000 x 10000
۱	0.56	0.60	0.63	0.62
۲	0.83	1.09	1.12	1.13
۴	1.11	1.51	1.95	1.93
۸	1.25	1.86	2.24	2.26

نتایج روش دوم)

تعداد نخ	اندازه ماتریس (موازی)			
	512 x 512	2048 x 2048	8192 x 8192	10000 x 10000
۱	0.001000	0.031000	0.444000	0.667000
۲	0.001000	0.017000	0.229000	0.346000
۴	0.001000	0.014000	0.129000	0.200800
۸	0.002000	0.010000	0.121000	0.187700

تعداد نخ	اندازه ماتریس (موازی)			
	512 x 512	2048 x 2048	8192 x 8192	10000 x 10000
۱	1	0.50	0.63	0.62
۲	1	0.91	1.10	1.08
۴	1	1.11	1.96	1.86
۸	0.5	1.56	2.09	1.99

گزارش)

کدها بر روی PC شخصی آزمایش شده‌اند.

برای کد بخش اول به این صورت عمل شده:

```
void add(DataSet dataSet, int numberOfThreads) {
    int i, j;
    #pragma omp parallel for private(j)
    for (i = 0; i < dataSet.n; i++) {
        for (j = 0; j < dataSet.m; j++) {
            dataSet.C[i * dataSet.m + j] = dataSet.A[i * dataSet.m + j] + dataSet.B[i * dataSet.m + j];
        }
    }
    #pragma opm barrier
}
```

که از دستور pragma omp parallel استفاده شده.

برای کد بخش دو به این صورت داریم:

```
void add(DataSet dataSet, int numberOfThreads) {
    int i, j, k, iStart, iEnd, jStart, jEnd;
    int chunksInRow = sqrt(NUMBER_OF_SQUARES);
    int cellsInChunk = dataSet.n / chunksInRow;
    #pragma omp parallel for private(i, j, iStart, iEnd, jStart, jEnd)
    for (k = 0; k < NUMBER_OF_SQUARES; k++) {
        iStart = floor(k / chunksInRow) * cellsInChunk;
        iEnd = iStart + cellsInChunk;
        jStart = cellsInChunk * (k % chunksInRow);
        jEnd = jStart + cellsInChunk;
        for (i = iStart; i < iEnd; i++) {
            for (j = jStart; j < jEnd; j++) {
                dataSet.C[i * dataSet.m + j] = dataSet.A[i * dataSet.m + j] + dataSet.B[i * dataSet.m + j];
            }
        }
    }
    #pragma opm barrier
}
```

مقدار NUMBER_OF_SQUARES یک ثابت است که مقدار آن 64 می‌باشد. یعنی در کل ماتریس به 64 مربع تقسیم می‌شود که هر کدام به یک نخ داده می‌شوند.

همانطور که می‌بینیم حالت تقسیم سطری همیشه بهتر از تقسیم مربعی عمل می‌کنیم؛ البته به جز برخی استثناءها که اندازه‌ی ماتریس بزرگ نیست. در کل شاید بتوان گفت به خاطر locality این نتایج حاصل می‌شود؛ زیرا در تقسیم سطری از locality در cache بهره بیشتری می‌بریم تا به صورت تقسیم مربعی.

وقتی تعداد نخ‌ها ۱ است، عملکرد بدتری نسبت به حالت سریال داریم. این به خاطر سربار اضافه‌ی openmp است که هیچ فایده‌ای برای ما ندارد. گاهی اوقات در حالت ۲ نخی هم وقتی ماتریس کوچک است ممکن است عملکرد بدتری داشته باشیم.

در کل وقتی ماتریس کوچک داریم، نمی‌توانیم تسریع زیادی داشته باشیم و در حالت تعداد نخ کم حتی بدتر هم عمل می‌کند.

کدها همگی در پوشه‌ی code قرار گرفته شده‌اند.