Kamil Karczewski
194976
Introduction to high performance computing

Mnożenie macierzy przez wektor

1. Co zostało zrobione.
    a. Zaimplementowane został algorytm mnożenia macierzy przez wektor, z obsługą macierzy niekwadratowej.
    b. Zrównoleglenie z użyciem openMP.
    c. Wczytanie macierzy z pliku dane.txt, wektora z pliku wektor.txt
    d. Tablice dynamiczne
    e. Rozmiar tablic przekazywany do programu przy uruchomieniu jako parametr z odczytywaniem poprzez funkcję atoi.
    f. Wynik przekierowany do pliku poprzez bashowe >>
    g. Czas mierzony przy użyciu biblioteki time.h
2. Dane
    a. Dane generowane przez osobne programy makeA.c oraz makeB.c odpowiednio dla macierzy i wektora. Macierz sprowadzona do tablicy jednowymiarowej.
3. Algorytm

$$c[0]=a[0][0]*b[0]+a[0][1]*b[1]+...+a[0][j]*b[j]+...+a[0][n]*b[n]$$
$$c[1]=a[1][0]*b[0]+a[1][1]*b[1]+...+a[0][j]*b[j]+...+a[0][n]*b[n]$$
$$c[i] =a[i][0] *b[0]+a[i][1] *b[1]+...+a[i][j] *b[j]+...+a[0][n]*b[n]$$
$$...$$
$$c[n]=a[n][0]*b[0]+a[n][1]*b[1]+...+a[n][j]*b[j]+...+a[n][n]*b[n]$$

a) Implementacja:

```
for(i=0;i<sizeA;i++){
        for(j=0;j<sizeB;j++){
                C[i]+=A[i*sizeA+j]*B[i];
        }
}
```

4. Czasy
    a. Lokalnie (Ubuntu 1gb RAM, Intel I5 CPU M450 2,40GHz na VBox udostępniony jeden rdzeń)

Wielkość	Input	Mnożenie	Output	IO	Razem
100x100	0,000981462	0,000054407	0,000023761	0,001005223	0,001059630
500x500	0,029995384	0,001388907	0,000084745	0,030080129	0,031469036
1000x1000	0,122021044	0,005667555	0,000200637	0,122221681	0,127889236
2500x2500	0,694614748	0,044070926	0,000390743	0,695005491	0,739076417
4096x4096	1,918137298	0,120950440	0,000823709	1,918961007	2,039911447
10000x10000	11,232813224	0,684097731	0,001644608	11,23445783	11,918555563

    b. Sigma

Wielkość	Input	Mnożenie	Output	IO	Razem
100x100	0,002440813	0,000137727	0,000115936	0,002556749	0,002694476
500x500	0,029698046	0,000439371	0,002362941	0,032060987	0,032500358
1000x1000	0,117277414	0,005438469	0,001088966	0,118366380	0,123804849
2500x2500	0,713742510	0,006411289	0,003083425	0,716825935	0,723237224
4096x4096	1,940436634	0,020692716	0,005284824	1,945721458	1,966414174
10000x10000	-1,798477353	0,135875664	0,015557983	----	-----

c. XeonPhi

Wielkość	Input	Mnożenie	Output	IO	Razem
100x100	0,0103206460	0,264045832	0,002379362	0,012700008	0,276745840
500x500	0,0253444318	0,253444318	0,007495816	0,248454529	0,501898847
1000x1000	0,9596143600	0,264811839	0,029963821	0,989578181	1,254390020
2500x2500	5,9835809200	0,261619442	0,020116397	6,003697317	6,265316759
4096x4096	16,045903134	0,250894567	0,037585028	16,083488162	16,334382729
10000x10000	95,660705670	0,278965640	0,084627540	95,745333210	96,024298850

5. Podsumowanie.
    a. Najwięcej czasu na Xeonie zajmuje wczytanie danych.
    b. Dla małych ilości danych nie zyskamy, czas obliczeń jest mniej więcej stały, natomiast czas całkowity jest zawyżany przez wczytywanie.
    c. Dla naprawdę ogromnych danych zyskamy na czasie obliczeń, co w kontekście dużych ilości danych może znacząco rzutować na różnicę w czasie całkowitym.