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	0,001152692	0,000103379	0,000025073	0,001177765	0,001281144
500x500	0,024916383	0,002024540	0,000098443	0,025014826	0,027039366
1000x1000	0,114484919	0,009416798	0,000228047	0,114712966	0,124129764
2500x2500	0,694053981	0,059374029	0,000473264	0,694527245	0,753901274
4096x4096	1,871750102	0,163630426	0,000711727	1,872461829	2,036092255
10000x10000	11,191873270	0,924134313	0,001471494	11,19334476	12,117479077

    b. Sigma

Wielkość	Input	Mnożenie	Output	IO	Razem
100x100	0,003163674	0,000222936	0,000027378	0,003191052	0,003413988
500x500	0,025260964	0,003489410	0,000088000	0,025348964	0,028838374
1000x1000	0,099639698	0,001276628	0,000163290	0,099802988	0,101079616
2500x2500	0,625817502	0,014041590	0,000389784	0,626207286	0,640248876
4096x4096	1,700637488	0,016826932	0,000679276	1,701316764	1,718143696
10000x10000	10,262963328	0,124806652	0,002981176	10,2659445	10,390751156

c. XeonPhi

Wielkość	Input	Mnożenie	Output	IO	Razem
100x100	0,010417953	0,262786775	0,000266027	0,010683980	0,273470755
500x500	0,241368815	0,243845823	0,000911316	0,242280131	0,486125954
1000x1000	0,960995524	0,263749665	0,001789778	0,962785302	1,226534967
2500x2500	5,983963589	0,260576130	0,004485676	5,988449265	6,249025395
4096x4096	16,05214374	0,252481041	0,006805187	16,058948931	16,311429972
10000x10000	95,652036486	0,257191213	0,017520764	95,66955725	95,926748463

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.