Mateusz Łąpieś (mateusz.lapies@student.pk.edu.pl)

## Sprawozdanie – Laboratorium 1

1. Pominięty w sprawozdaniu – załączony kod zawiera katalog roboczy.
2. Każda strategia wspomniana w definicji laboratoriów została zaimplementowana w postaci osobnej funkcji nazwanej według konwencji strategy\_(litera reprezentująca strategię)  
   Przykładowa implementacja procedury strategy\_a:

void strategy\_a(int iterations) {

int i;

#pragma omp parallel for schedule(static,3)

for(i=0;i<iterations;i++){ printf("thread %d, index %d\n",omp\_get\_thread\_num(),i); }

}

1. Poniżej listingi otrzymanych wpisów do konsoli dla poszczególnych przypadków:

thread 1, index 3

thread 1, index 4

thread 2, index 6

thread 2, index 7

thread 2, index 8

thread 0, index 0

thread 0, index 1

thread 0, index 2

thread 0, index 12

thread 0, index 13

thread 0, index 14

thread 3, index 9

thread 3, index 10

thread 3, index 11

thread 1, index 5



thread 3, index 12

thread 0, index 0

thread 0, index 1

thread 0, index 2

thread 0, index 3

thread 3, index 13

thread 3, index 14

thread 2, index 8

thread 2, index 9

thread 2, index 10

thread 2, index 11

thread 1, index 4

thread 1, index 5

thread 1, index 6

thread 1, index 7



thread 0, index 3

thread 0, index 4

thread 0, index 5

thread 0, index 9

thread 0, index 10

thread 0, index 11

thread 0, index 12

thread 0, index 13

thread 0, index 14

thread 2, index 6

thread 2, index 7

thread 2, index 8

thread 3, index 0

thread 3, index 1

thread 3, index 2



thread 0, index 0

thread 0, index 2

thread 0, index 3

thread 0, index 4

thread 0, index 5

thread 0, index 6

thread 0, index 7

thread 0, index 8

thread 0, index 9

thread 0, index 10

thread 0, index 11

thread 0, index 12

thread 0, index 13

thread 0, index 14

thread 2, index 1

1. W tym podpunkcie zgodnie z poleceniem została wybrana operacja dodawania. Oczywiście w implementacji tuż przed uruchomieniem poszczególnych przypadków pobierany jest aktualny czas za pomocą funkcji omp\_get\_wtime(), natomiast po zakończeniu działania procedury tworzony jest wpis do konsoli zawierający różnicę zapisanej wartości czasu od aktualnie zwracanej wartości z funkcji omp\_get\_wtime().  
   Poniżej wykres zależności czasu działania zaimplementowanego algorytmu względem strategii dystrybucji zadań na poszczególne wątki.
2. Poniżej przedstawiona jest implementacja algorytmu dla obu przypadków (bez redukcji oraz z nią) oraz wykres zależności czasu pracy algorytmu od uruchomionego przypadku. Ilość iteracji (5000) jest przekazywane w postaci parametru dla obu procedur. Definicja ilości iteracji oraz mierzenie czasu pracy algorytmu są przeprowadzane w głównej procedurze (main).

void strategy\_non\_reduction(int iterations) {

int i; double sum = 0;

#pragma omp parallel for

for(i=0;i<iterations;i++){ sum += pow(2,2); }

}

void strategy\_reduction(int iterations) {

int i; double sum = 0;

#pragma omp parallel for reduction(+:sum)

for (i = 0; i < iterations; i++) { sum += pow(2, 2); }

}

1. Pominięte w sprawozdaniu – przykładowy program
2. Poniżej można znaleźć implementację algorytmu mnożenia dwóch macierzy o wymiarach NxM oraz MxP. Macierz wynikowa ma wymiary NxP.

for(i = 0; i < N; i++) {

for(j = 0; j < P; j++) {

for(k = 0; k < M; k++) {

output[i][j] += main[i][k] \* multi[k][j];

}

}

}