Anna Plust, 36148
Równoległe programowanie inkrementacyjne

Laboratorium 2: Dyrektywa parallel

Zadanie 1

Podczas zadania utworzono 12 programów, z których każdy implementował inną kombinację następujących mechanizmów wykorzystywanych do zrównoleglania obliczeń:

- zmienna środowiskowa (OMP_NUM_THREADS)
- klauzula dyrektywy parallel (num_threads(<int>))
- funkcja biblioteki OpenMP (omp_set_num_threads(<int>))
- mechanizm wątków dynamicznych po sprawdzeniu czy ten mechanizm jest zaimplementowany (set_dynamic(<bool>), get_dynamic())
- klauzula if (if())
- domyślna implementacja - maksymalna dostępna ilość wątków w procesorze

Na podstawie wykonanego eksperymentu można utworzyć hierarchię komend wg pierwszeństwa (od najwyższego do najniższego):

1. klauzula if (if())
2. klauzula dyrektywy parallel (num_threads(<int>))
3. funkcja biblioteki OpenMP (omp_set_num_threads(<int>))
4. zmienna środowiskowa (OMP_NUM_THREADS)
5. domyślna implementacja - maksymalna dostępna ilość wątków w procesorze
6. mechanizm wątków dynamicznych po sprawdzeniu czy ten mechanizm jest zaimplementowany (set_dynamic(<bool>), get_dynamic())

Zadanie 2

Napisać program w OpenMP do znalezienia sumy wszystkich elementów wektora z zastosowaniem klauzuli reduction.

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

int main(int argc, char *argv[])
{
   int sum = 0;
   int size = 5;
   int vector[5] = {1, 2, 3, 4, 5};

   #pragma omp parallel for reduction(+ : sum)
       for (int i = 0; i < sizeof(vector)/sizeof(vector[0]); i++) {
           sum += vector[i];
       }

   printf("sum  = %d\n", sum);
}
```

Zadanie 3

Napisać program do porównania wyników generowanych przez program sekwencyjny i program równoległy i zastosować go do sprawdzenia poprawności programu równoległego (zadanie 2) dla ograniczonego rozmiaru problemu.

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

int main(int argc, char *argv[])
{
   int size = 200;
   int vector[200];

   for (int i = 0; i < size; i++) {
       vector[i] = i+1;
   }

   int sumParallel = 0;
   #pragma omp parallel for reduction(+ : sumParallel)
       for (int i = 0; i < sizeof(vector)/sizeof(vector[0]); i++) {
           sumParallel += vector[i];
       }

   int sumSequential = 0;
   for (int i = 0; i < sizeof(vector)/sizeof(vector[0]); i++) {
       sumSequential += vector[i];
   }

   printf("sumParallel = %d\n", sumParallel);
   printf("sumSequential = %d\n", sumSequential);
}
```