Z2	Przemysław Kleszcz, Krzysztof Tatar	05/04/2018

Zadanie 1 - Rozmycie Gaussa w OpenMP

Celem zadania było napisanie programu, który rozmywa zadane zdjęcie za pomocą algorytmu Gaussa z maską o wymiarach 5x5. Do poprawnego działania programu należy podać trzy argumenty wejściowe. Są to *n -liczba wątków*, *input_image - ścieżka do pliku obrazu w formacie JPEG* i *output_image - ścieżka do pliku wyjściowego obrazu w formacie JPEG* .Poniżej przedstawiony jest rysunek zastosowanej filtracja na podstawie filtra o masce 5 x 5 wraz z wagami, które są wykorzystywane podczas przeprowadzania obliczeń.

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Rysunek 1: Wykorzystana maska Gaussowska

Poniżej zaprezentowano główną funkcję programu odpowiedzialną za zrównoleglanie obliczeń.

```
1  void gaussian(int weight,Mat picture,Mat pictureNew,int kernel[][5],int threadsNumber)
2  {
3     int y, x;
4     int r, g, b;
5
6  #pragma omp parallel shared(picture,pictureNew) num_threads(threadsNumber) private(r,g,b,
       y,x)
7     {
8  #pragma omp for schedule(guided,1)
9        for (y = 0; y < picture.rows; y++)
10       {
11          for (x = 0; x < picture.cols; x++)
12          {
13          ...
14          ...
15          ...
16          }
17       }
18    }
19 }
```

W powyższym programie został zastosowany algorytm filtrowania za pomocą którego posiadamy możliwość rozmycia podanego obrazka. Do zrównoleglenia głównej pętli for została użyta dyrektywa OpenMP.Wyraz omp jest słowem kluczowym biblioteki. Dyrektywa parallel, wskazuje kompilatorowi obszar kodu, który będzie zrównoleglony.

Zrównoleglenie głównej pętli for:

```
1 #pragma omp for
```

Kolejna dyrektywa - for - informuje kompilator, że zrównoleglana będzie pętla typu for.

Określnie, które zmienne będą wspólne shared, a które prywatne private:

```
1 shared(picture,pictureNew) private(r,g,b,y,x)
```

Zmienne wspólne są dostępne dla każdego wątku, natomiast do danej zmiennej prywatnej ma dostęp tylko jeden określony wątek. W naszym programie zmiennymi prywatnymi są: r,g,b,y,x, a zmiennymi wspólnymi picture,pictureNew.

Dyrektywa schedule:

```
1 schedule(guided, 1)
```

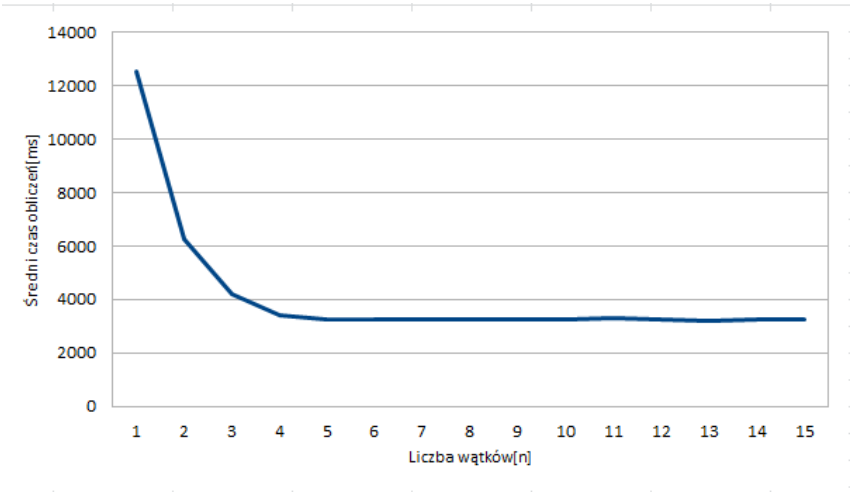
Za pomocą polecenia schedule możemy kontrolować sposób w jaki OpenMP przydziela iteracje dostępnym wątkom. Parametr guided powoduje przypisanie do każdego wąktu odpowiednio dużego fragmentu następujących po sobie iteracji. Rozmiar fragmentu zmniejsza się wykładniczo, wraz z każdym pomyślnym przypisaniem, do minimalnego rozmiaru zdefiniowanego w parametrze „chunk". Parametr „chunk" jest ustawiony na 1, oznacza to że rozmiar każdego z początkowych zbiorów jest opisany wyrażeniem:

liczba_nieprzydzielonych_iteracji/liczba_wątków

Dyrektywa num_threads:

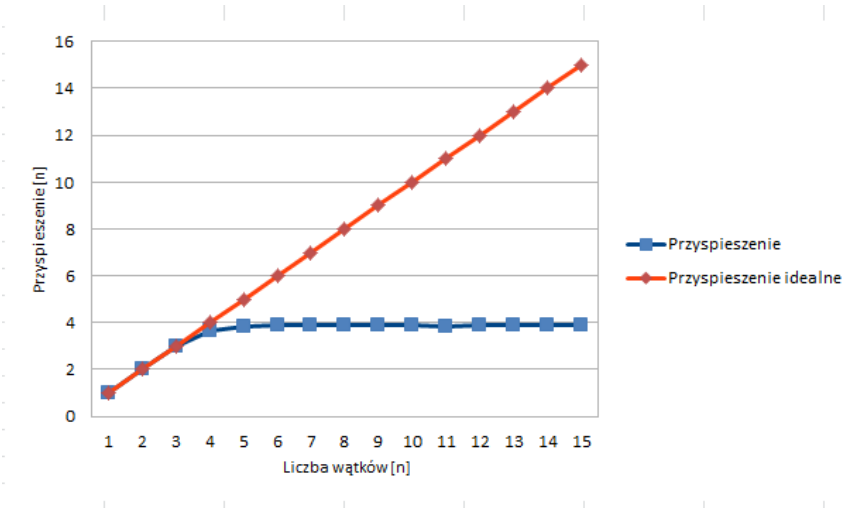
```
1 num_threads(threadsNumber)
```

Za pomocą dyrektywy num_threads określamy, ile wątków ma być użytych do zrównoleglenia pętli for.



Rysunek 2: Wykres średniego czasu obliczeń

Z powyższego rysunku można wywniokoswać, że średni czas obliczeń dynamicznie malał dla pierwszych 4 wątków. Następnie dla kolejnych wątków średni czas jest ustabilizowany. Poniższy rysunek jest dobrym przykładem opisanego zjawiska.



Rysunek 3: Wykres przyspieszenia

Powyższe zadanie zostało zrównoleglone dzięki bibliotece OpenMP. Dzięki zastosowaniu dyrektyw biblioteki udało się uzyskać czterokrotne przyspieszenie dla 4 wątków. Dla kolejnych 11 wątków, w tym przykładzie nie wpłynęły znacząco na wydajność.