

Lab 1

środa, 13 października 2021 17:50

Oscar Teeninge

Algorytmy równoległe 16:15 cew.

$$\textcircled{1} \quad \frac{\delta^2 V}{\delta x^2} + \frac{\delta^2 V}{\delta y^2} = 0$$

Pредставление проблема

Używamy metod równic skończonych MRS

$$h = \frac{b-a}{n} \quad k = \frac{c-d}{m}$$

Dzięki temu możemy podzielić przedział $[a, b]$ na n równych części o szerokości h oraz przedział $[c, d]$ na m równych części o szerokości k .

$$x_i = a + ih \quad i = 0, 1, \dots, n$$

$$y_j = c + jk \quad j = 0, 1, \dots, m$$

Dla zmiennej x :

$$\frac{\delta^2 V(x_i, y_j)}{\delta x^2} = \frac{V(x_{i+1}, y_j) - 2 \cdot V(x_i, y_j) + V(x_{i-1}, y_j)}{h^2}$$

Dla zmiennej y :

$$\frac{\delta^2 V(x_i, y_j)}{\delta y^2} = \frac{V(x_i, y_{j+1}) - 2 \cdot V(x_i, y_j) + V(x_i, y_{j-1})}{k^2}$$

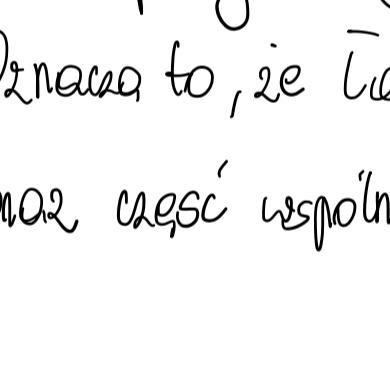
Podstawiając do wzoru i przyjmując $V(x_i, y_j) = V_{i,j}$

$$(V_{i+1,j} + V_{i-1,j}) \frac{1}{h^2} + (V_{i,j+1} + V_{i,j-1}) \frac{1}{k^2} = \left(\frac{2}{h^2} + \frac{2}{k^2} \right) V_{i,j}$$

Zauważmy, że można to przedstawić w iteracyjnej formie

$$V_{i,j}^{(n+1)} - V_{i,j}^{(n)} = \frac{1}{h^2} (V_{i+1,j}^{(n)} + V_{i-1,j}^{(n)}) + \frac{1}{k^2} (V_{i,j+1}^{(n)} + V_{i,j-1}^{(n)}) - \left(\frac{2}{h^2} + \frac{2}{k^2} \right) V_{i,j}^{(n)}$$

Problem zadania



- potencjał V

- potencjał 0

$k = h$ - ponieważ kwadrat

$n = m$

Oznacza to, że $V_{0,j} = V_{i,0} = V_{n,j} = V_{i,n} = 0$

Zauważmy, że u nas $h = k$, ponieważ $b - a = d - c$. Oznacza to:

$$V_{i,j}^{(n+1)} - V_{i,j}^{(n)} = (V_{i,j+1}^{(n)} + V_{i,j-1}^{(n)} + V_{i+1,j}^{(n)} + V_{i-1,j}^{(n)}) \cdot \frac{1}{h^2}$$

Dekompozycja



Podział powinien być możliwie jak najprostszy i równomierny, oraz powodować jak najmniejszą potrzebę synchronizacji.

Moim pomysłem jest podział na równne kwadraty.

Oznacza to, że iteracja będzie zrealizować podział

ale części wspólna jest linijna zależna od liczby podziałów.

Komunikacja



Część wspólna będąca na granicach między obszarami

wymaga synchronizacji. Koniec i początek iteracji również

musi być synchronizowane, w przeciwnym wypadku może zdarzyć się

sytuacja, że pomimo synchronizacji wartości, inny wątek następującej iteracji

zapołocie zbyt szybko. W przypadku operacji części wspólnej można zastosować

operacje atomowe. Moim pomysłem jest przechowywanie kopii macierzy i ona

bedzie reprezentować poprzednią iterację, a wartości aktualnej iteracji

zapisywać będą w innej macierzy. Na koniec iteracji nastąpi

stępnianie. Rozwiązuje to problem w którym brakują się warunki

zgodności problemu / BLOCKSIZE.

Scalanie

Nie ma jednego optymalnego podziału. W przypadku obliczeń na

cpu stabilizujemy się odwzorowaniem 1:1, natomiast dla planowania wykonywania

obliczenia na gpu - jeszcze nie wiem na jakiej technice będzie wykonywane

obliczenia, jednakże kiedy obliczeń zależy linijnie od rozmiaru problemu.

Mozna zauważyć, że BLOCKSIZE brakie 8x8, co oznacza kiedy gridów to

rozmiar problemu / BLOCKSIZE.

Działanie

Każdy grid otrzymuje jedno zadanie iteracji na dane.

Konkrementy z pamięci wspólnieowej gpu. Posiadamy konie

tablicy, na nich wykonyujemy operacje atomowe, a gory

wszystkie gridy zadanego - zapisujemy kopie jako wynik,

i w nowej iteracji powtarzamy proces.

kolejna iteracja

ALOKACJA NA CPU \rightarrow ALOKACJA NA GPU \rightarrow KOPIOWANIE $M_{CPU} \rightarrow M$ $M_{CPU} \rightarrow M$ $M_{CPU} \rightarrow M$ M_{CPU}