Równanie przewodnictwa cieplnego w 2D przy użyciu przetwarzania równoległego MPI

B. Nowak, M. Wawrzak, M. Lachman

12 lutego 2016

1 Wstęp

Celem projektu było zapoznanie się z biblioteką MPI oraz przy jej użyciu napisanie programu przewodnictwa cieplnego w 2D w programie w języku C. Otrzymane wyniki należało zwizualizować przy pomocy skryptu napisanego w języku python i bibliotece VPython, która odpowiada za wyświetlanie grafiki w tym języku.

2 Wykorzystane Narzędzia i biblioteki.

2.1 Biblioteka MPI

Message Passing Interface (MPI) - protokół komunikacyjny będący standardem przesyłania komunikatów pomiędzy procesami programów równoległych działających na jednym lub więcej komputerach. Interfejs ten wraz z protokołem specyfikuje, jak jego elementy winny się zachowywać w dowolnej implementacji. Celami MPI są wysoka jakość, skalowalność oraz przenośność.

Najbardziej znaną implementacją MPI jest MPICH, pochodzący z Argonne National Laboratory i rozwijany przez grupę pracowników działu matematyki i informatyki tej instytucji. Dostępna jest wersja zarówno na platformy UNIX-owe, jak i Windows.

MPI realizuje model przetwarzania współbieżnego zwany MIMD, a dokładniej SPMD. Zakłada on, że ten sam kod źródłowy wykonuje się jednoczenie na kilku maszynach i procesy mogą przetwarzać równocześnie różne fragmenty danych, wymieniając informacje przy użyciu komunikatów. Takie podejście ma wiele zalet, z których najbardziej spektakularną jest chyba możliwość współbieżnych obliczeń wykonywanych na maszynach o zupełnie różnych architekturach (np. Linux-x86 oraz Solaris-Sparc).

2.2 VPython

VPython jest to Python dodatkowo z modułem graficznym 3D zwanym Visual. VPython pozwala użytkownikowi na tworzenie obiektów w przestrzeni 3D. Pozwala to na łatwe tworzenie prostych wizualizacji, pozwalając programistom skupić się na obliczeniowych aspektach ich programów. Prostota biblioteki VPython stała się narzędziem do ilustracji prostych zjawisk fizycznych, zwłaszcza w środowisku edukacyjnym.

2.3 Makefile

Make - program powłoki systemowej automatyzujący proces kompilacji programów, na które składa się wiele zależnych od siebie plików.

Program przetwarza plik reguł Makefile i na tej podstawie stwierdza, które pliki źródłowe wymagają kompilacji. Zaoszczędza to wiele czasu przy tworzeniu programu, ponieważ w wyniku zmiany pliku źródłowego kompilowane są tylko te pliki, które są zależne od tego pliku. W naszym projekcie utworzyliśmy plik Makefile dzięki czemu, przy zmianach w plikach nie ma potrzeby kompilacji całego projektu.

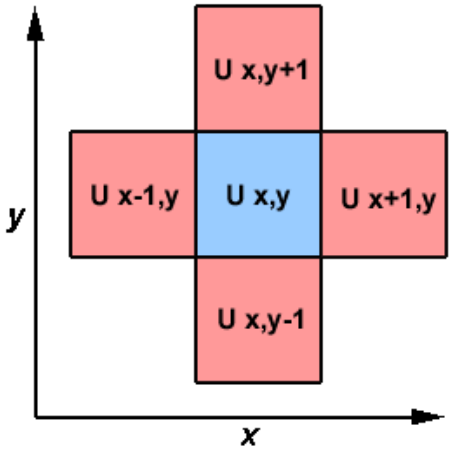
3 Prosty model przepływu ciepła

Równanie ciepła opisuje zmianę temperatury w czasie, biorąc pod uwagę początkowy rozkład temperatury i warunków brzegowych. W naszym projekcie zakładamy istnienie siatki temperatur, w której zawsze na brzegach występuje temperatura 0 stopni, natomiast w środku temperatura może się zmieniać zależnie od wartości elementu sąsiada. Wartości te otrzymujemy według następujących wzorów:

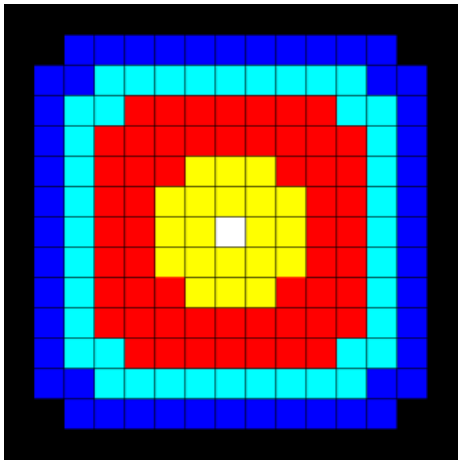
$$
\begin{aligned}
U_{x,y} = {} & U_{x,y} \\
& + C_x * (U_{x+1,y} + U_{x-1,y} - 2 * U_{xy}) \\
& + C_y * (U_{x,y+1} + U_{x,y-1} - 2 * U_{x,y})
\end{aligned}
$$

4 Wykonanie projektu

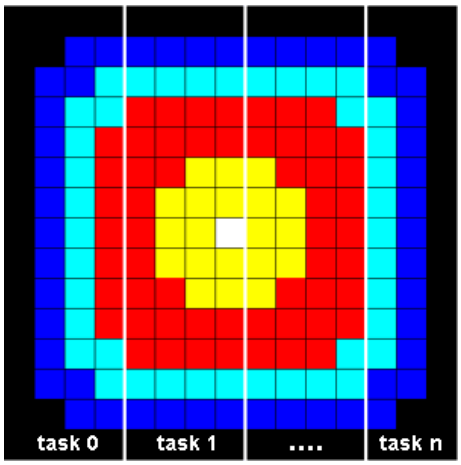
Część obliczeniowa projektu została wykonana w języku C z wykorzystaniem biblioteki do obliczeń równoległych MPICH. Następnie otrzymane wyniki przedstawiliśmy w formie graficznej przy pomocy skryptów w języku Python oraz biblioteki VPython.



Rysunek 1: Sposób ozaczenia kolejnych elementów siatki wymaganych do obliczeń.



Rysunek 2: Początkowy rozkład temperatur siatki. Wartości na brzegach w każdej chwili równają się 0.



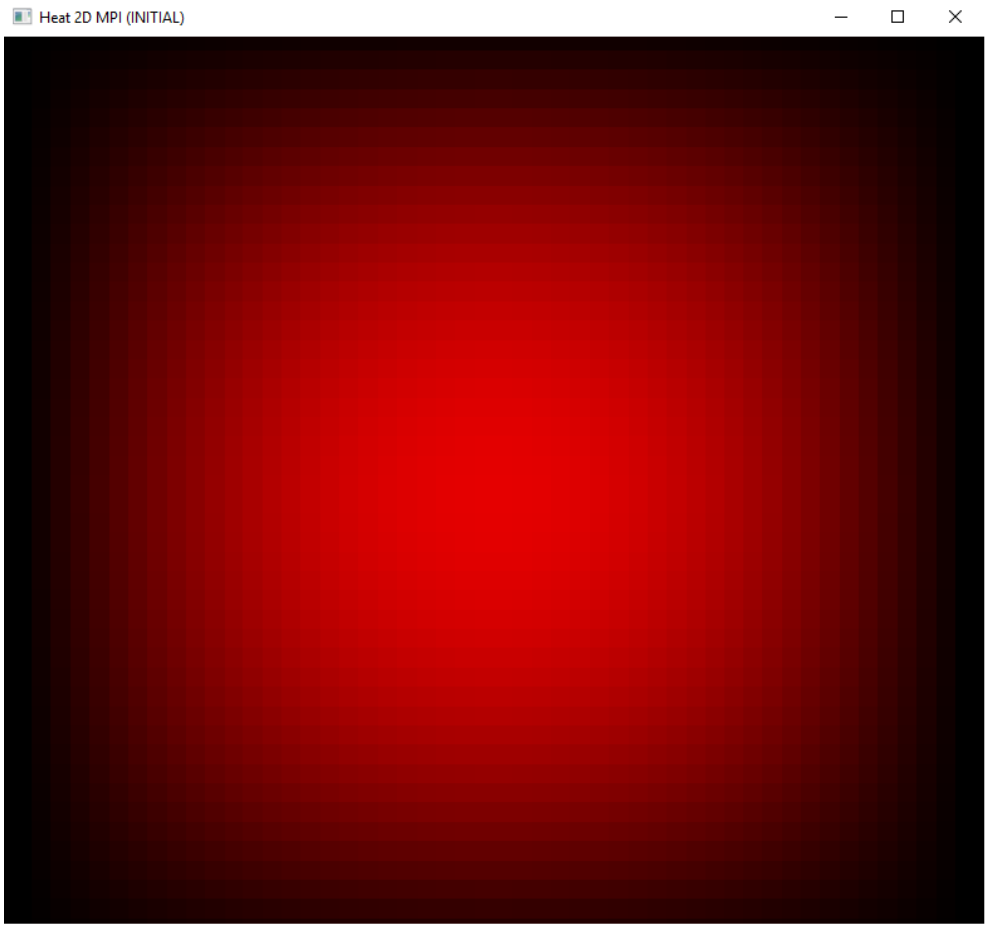
Rysunek 3: Sposób rozdziału danych na odpowiednią liczbę wątków.

4.1 Główny program heat.c

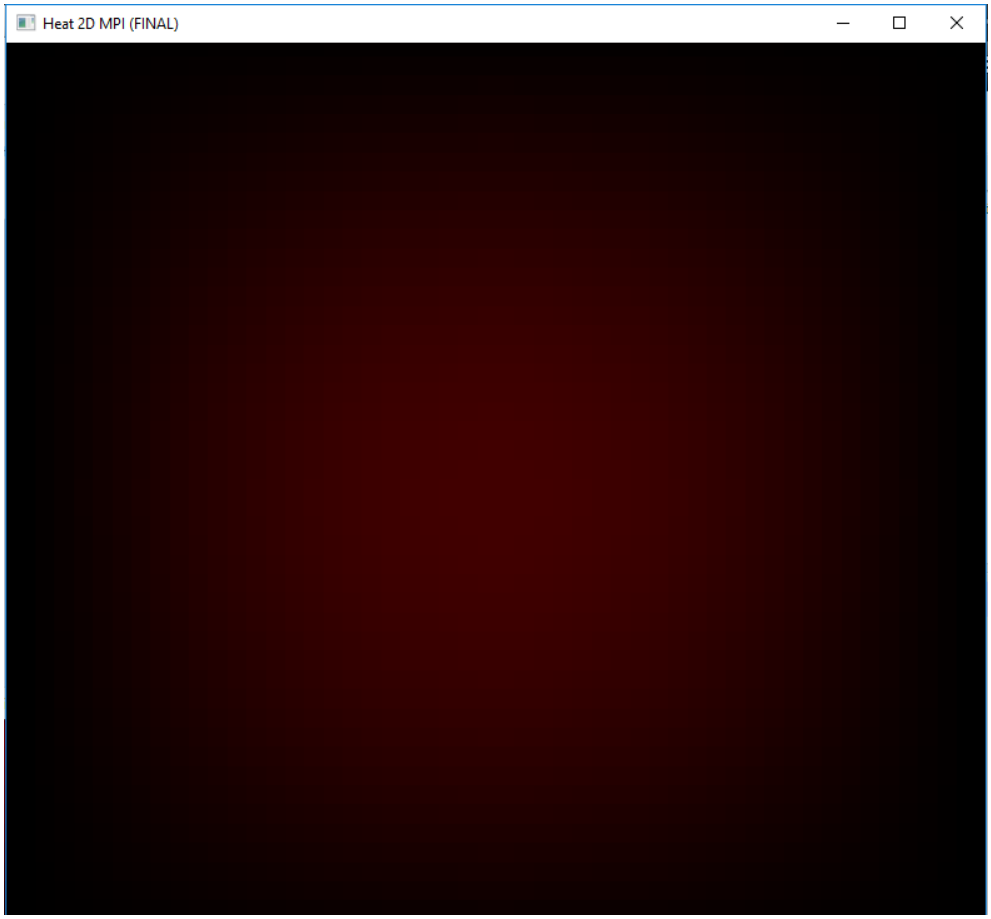
Na początku definiujemy zmienne wykorzystywane w programie. Następnie definiujemy Nx i Ny czyli wymiary naszej siatki, liczbę kroków oraz ilość procesów. W kolejnym kroku uruchamiana jest funkcja inicjująca siatkę w zależności od podanych wcześniej parametrów. Kolejnym krokiem jest wywołanie biblioteki MPI poprzez funkcję inicjującą środowisko wykonywania programu. Dopiero od momentu wywołania MPI_ Init można używać pozostałych funkcji biblioteki MPI. Następnie tworzymy procesy na jeden zarządzający oraz procesy wykonywujące. Funkcja, która pobiera ilość procesów to MPI_ Comm_ size. Potem poprzez funkcję MPI_ Send funkcja wysyła komunikaty do procesów. Procesy pośrednie liczą przepływ ciepła w przypisanych do nich sektorach w zależności od ich sąsiadów. Poprzez funkcję MPI_ Comm_ rank pobierany jest numer aktualnego procesu. Proces zarządzający "przydziela" zadania procesom oraz po zakończeniu przez nie pracy zbiera wyniki i zwraca całościowe rozwiązanie problemu. Każdy proces wylicza wartość danego punktu z wartości jego najbliższych sąsiadów. W przypadku kiedy do wyliczenia wartości danego punktu, potrzeba informacji o jego sąsiedzie, który znajduje się w obszarze innego wątku, komunikuje się on ze swoim sąsiadem i otrzymuje informację zwrotną dotyczącą wartości danego punktu. Informacje zebrane poprzez procesy umieszczane są w buforze za pomocą funkcji MPI_ Recv. Następnie uruchamiana jest funkcja uaktualniająca siatkę i cały proces następuje od nowa. Funkcją, która zwalnia zasoby i przygotowuje program do zamknięcia jest MPI_ Finalize.

4.2 Program odpowiedzialny za wizualizacje danych

Program ten miał za zadanie wczytanie danych z programu głównego heat.c. Następnym krokiem było utworzenie z tych danych tablicy dwuwymiarowej. Znormalizowaliśmy dane do wartości maksymalnych. Z tak przygotowanych danych stworzyliśmy mapę wartości, następnie przerysowaliśmy w postaci kwadratów, każdy z nich był pokolorowany zgodnie z jego wartością. Na odpowiedni odcień pomiędzy czarnym a czerwonym. W ten sposób otrzymaliśmy dwa okna przedstawiające dwa stany układu początkowy i końcowy.



Rysunek 4: Siatka inicjująca. Początkowy rozkład temperatury T na siatce o wymiarach 50 x 50.



Rysunek 5: Siatka końcowa. Rozkład temperatury po 300 krokach czasowych.

4.3 Wnioski

Rozwiązaliśmy zagadnienie prostego przepływu ciepła, przy pomocy programowania równoległego z wykorzystaniem biblioteki MPI. Użycie wielu wątków znacznie przyspieszyło obliczenia w naszym programie. Praca przy tym projekcie pozwoliła nam zaznajomić się z funkcjami oferowanymi przez bibliotekę MPI, sposobami przekazywania danych za pomoc komunikatów oraz możliwościami jakie oferuje zarządzenie kolejnością wykonywanych procesów. Poniżej zamieszczamy wyniki wizualizacji przedstawiające początkowy i końcowy rozkład temperatury.