Dokumentacja

Systemy równoległe i rozproszone

Projekt 2 – Aplikacja równoległa UPC++

Temat: Problem oddziaływania wielu ciał (N-body problem)

Karol Rojek, Michał Zbrożek

1. Wprowadzenie teoretyczne i cele

Celem projektu było stworzenie aplikacji równoległej w wybranej technologii, która będzie rozwiązywać problem wielu ciał (N-body problem). Jest to zagadnienie mechaniki klasycznej polegające na obliczaniu toru ruchu ciał pewnego układu znając ich masy, prędkości i położenia początkowe. Zależą one od działających w układzie sił grawitacyjnych pomiędzy ciałami. Uproszczony schemat takiego układu 2D przedstawia Rys. 1. Aplikacja będzie obliczać analogiczny problem dla trzech wymiarów.

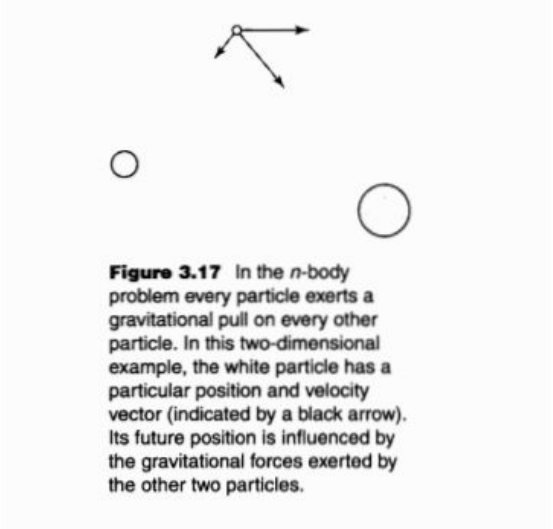


Figure 3.17 In the *n*-body problem every particle exerts a gravitational pull on every other particle. In this two-dimensional example, the white particle has a particular position and velocity vector (indicated by a black arrow). Its future position is influenced by the gravitational forces exerted by the other two particles.

Rys. 1 - Uproszczony schemat problemu wielu ciał 2D. Źródło: *M. J. Quinn, Parallel Programming in C with MPI and OpenMP, 2001*

Aby znaleźć położenia wszystkich ciał należy obliczyć działającą nań siłę wypadkową zgodnie z prawem powszechnego ciążenia (1).

(1) $$F^i = G\frac{m_1 m_2}{r^2}e^i,$$

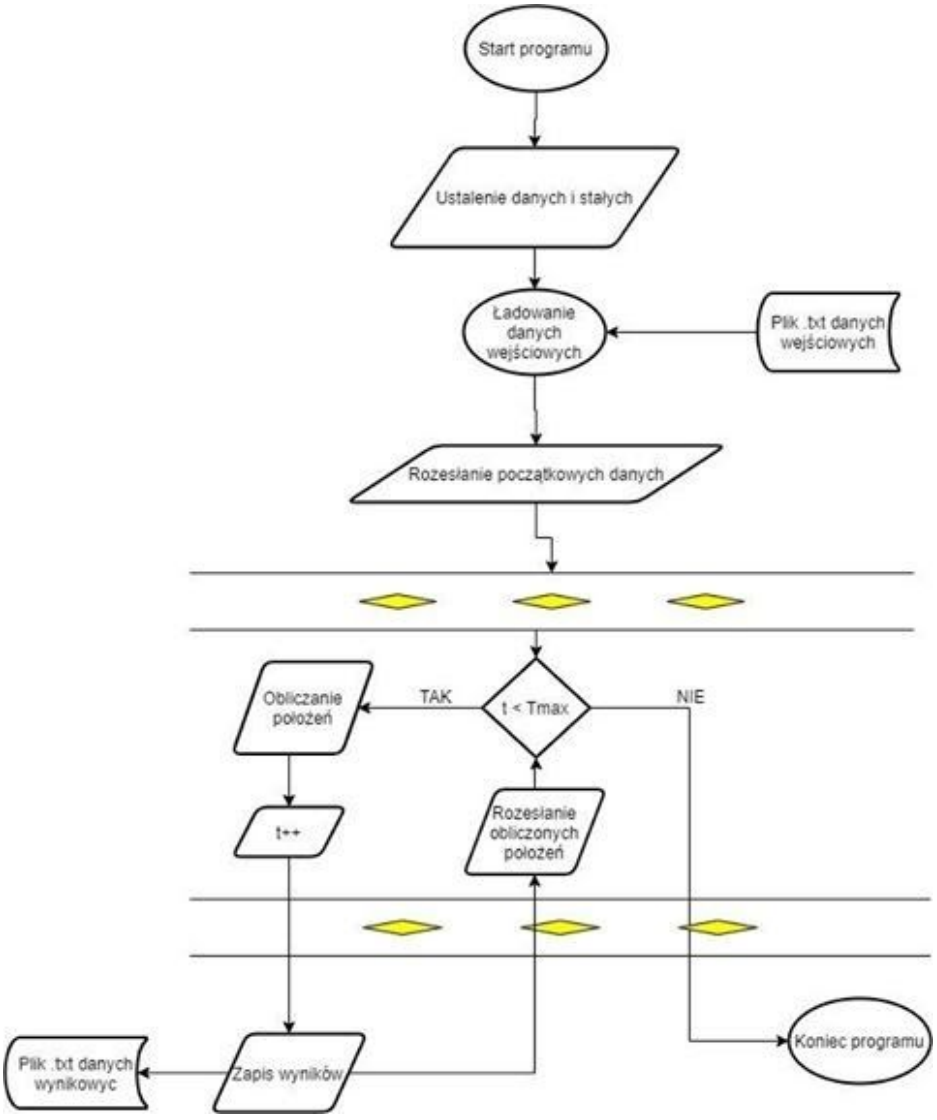
Następnie na podstawie tej siły obliczane jest przyspieszenie ciała, modyfikowany jest jego wektor prędkości, a na końcu położenie.

Ponieważ celem jest stworzenie aplikacji równoległej aby przyspieszyć wykonywane obliczenia, konieczna jest modyfikacja algorytmu problemu. Obliczenie nowego położenia danego ciała wymaga znajomości danych wszystkich pozostałych ciał - każdy z procesów musi posiadać cały zestaw danych, nie można go podzielić. Procesy mogą natomiast obliczać nowe pozycje tylko pewnego wybranego podzbioru ciał, a następnie wymieniać się wynikami. W ten sposób po jednym wykonaniu pętli algorytmu każdy z procesów zawiera dane o położeniu wszystkich ciał w następnym kroku czasowym.

2. Budowa i opis programu

Algorytm działania

Aplikacja obliczająca problem n-ciał stworzona jest w języku C++ z wykorzystaniem biblioteki UPC++. Schemat blokowy (Rys. 2) przedstawia uproszczony przebieg algorytmu.



Rys. 2 - Schemat blokowy aplikacji równoległej problemu n-ciał.

Program rozpoczyna działanie od ustawienia zmiennych potrzebnych do realizacji i określających działanie, takich jak krok czasowy i czas symulacji, oraz stałych matematycznych. Zainicjalizowane zostają też mechanizmy UPC++ za pomocą funkcji *init*, *rank_me* i *rank_n*.

Następnie proces główny programu odczytuje ilość symulowanych ciał z pliku wejściowego, oraz zapisuje ją pod globalnym adresem *totalObjectCountPtr*, skąd zostaje ona pobrana przez pozostałe procesy. Na jej podstawie każdy proces oblicza dla których ciął będzie prowadził symulację. Kolejnym krokiem jest wczytanie przez proces główny danych wejściowych, które są z kolei umieszczane w globalnej tabeli *dataVector*. Wszystkie procesy kopiują dane z *dataVector* do lokalnych tablic, aby ograniczyć operacje komunikacyjne.

W każdym z węzłów wywoływana jest pętla symulacji, która oblicza siły wypadkowe, przyspieszenia, oraz nowe położenia ciał. Krokiem czasowym pętli jest 1 minuta. Dane o nowych położeniach ciał są następnie przesyłane z powrotem do *dataVector*, oraz pobierane przez inne procesy, w celu ich aktualizacji. Co 10 minut położenia wszystkich ciał zapisywane są przez proces główny do pliku tekstowego. W ten sposób w pliku wyjściowym tworzy się historia położeń każdego z ciał, czyli tak zwana efemeryda.

Pętla kończy się po upływie jednego miesiąca, a po jej zakończeniu program zwalnia zasoby i kończy działanie funkcją *finalize*.

Funkcje zdefiniowane w programie

getObjectCount
Oblicza oraz zwraca ilość wpisów danych zawartych w pliku wejściowym.

readData
Czyta plik z danymi wejściowymi i zapisuje startowe położenia, prędkości i masy wszystkich ciał do przechowujących te dane tablicy globalnej *dataVector*.

splitData
Określa podział załadowanych danych pomiędzy węzły, umożliwia zrównoleglenie obliczeń.

saveData
Dodaje aktualne dane o położeniu każdego ciała do pliku wynikowego.

initializeVectors
Inicjalizuje lokalne wektory danych na podstawie wektora globalnego *dataVector*.

updateDataVector
Aktualizuje *dataVector* o nowo obliczone wartości położenia ciał.

updatePositionVectors
Aktualizuje lokalne wektory położeń na podstawie wartości z *dataVector*.

Dane wejściowe

Dane wejściowe programu to zestaw położeń, prędkości oraz mas wybranych 36 ciał niebieskich z dnia 01.01.2020 o godzinie 00:00. Położenie ciał zostało zaczerpnięte w postaci sferycznej z serwisu HORIZONS organizacji NASA z heliocentrycznego punktu widzenia, a następnie transformowane do postaci kartezjańskiej. Aby uniknąć sytuacji dzielenia przez 0 podczas konwersji Słońce zostało przesunięte o 1 metr, co nie ma znaczącego wpływu na obliczenia symulacji. Prędkości ciał zostały obliczone na podstawie różnicy w położeniu pomiędzy wymienioną datą a punktem dziesięć minut później.

Dane zapisane są w pliku *nbodydata.txt*, gdzie pierwsza linia stanowi opis danych, a każda kolejna to wpis mówiący o jednym ciele niebieskim.

Dane wyjściowe

Dane wyjściowe zapisywane są w pliku *resultdata.txt* w postaci podobnej do wejściowych, ale ograniczone są tylko do pozycji ciał. Do ich podglądu zalecamy użycie załączonego skryptu wizualizującego wykonanego w MATLAB, *vizualization.m*.

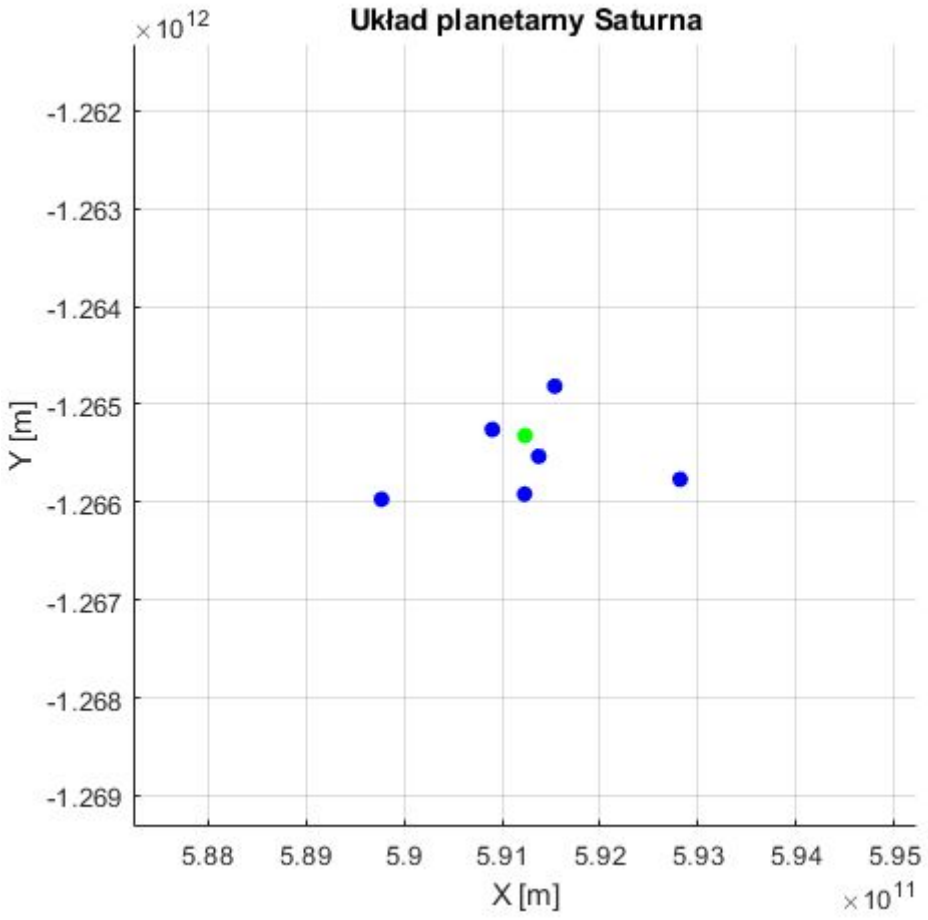
Obserwacje danych wyjściowych wskazują na istnienie pewnej niedokładności ze stanem rzeczywistym układu słonecznego - orbity księżyców różnych planet wydają się być bardziej ekscentryczne niż być powinno. Może to wynikać z ograniczonej do 3 miejsc po przecinku dokładności pobranych współrzędnych sferycznych.

3. Obsługa programu

Do programu dołączony jest plik *Makefile* udostępniający funkcjonalność kompilacji, uruchomienia i posprzątania po programie, za pomocą trzech poleceń, odpowiednio *make*, *make run*, *make clean*. Istnieje także polecenie *make all* wykonujące cały ten proces. Program nie kompiluje się na serwerze Taurus z powodu braku bibliotek CUDA, musi być kompilowany na stacjach roboczych. Program wymaga uprzedniego ustawienia zmiennych środowiskowych poleceniem:

source /opt/nfs/config/source_upcxx.sh

Dane zapisane w pliku *resultdata.txt* mogą być wykorzystane z dołączonym programem MATLAB *vizualization.m* do wizualizacji przebiegu orbit ciał poprzez stworzenie pliku wideo. Domyślnie skrypt pokazuje rzut układu planetarnego Jowisza na płaszczyznę XY, a krok czasowy to 180 minut. Rysunek 3 przedstawia zrzut ekranu z przykładowego wideo.



Rys. 3 - Prosta wizualizacja układu obliczonego w problemie wielu ciał.