AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Fizyki i Informatyki Stosowanej

Dokumentacja

Kamil Potoczny, Bartosz Piwnik

Systemy równoległe i rozproszone - aplikacja MPI

Monte Carlo - ruch na rondzie

Kraków, Kwiecień 2016
AGH, WFiIS, Informatyka Stosowana

1 Wstęp

Celem ninjeszego projektu było stworzenie aplikacji symulującej ruch pojazdów na rondzie z wykorzystaniem metody Monte-Carlo.

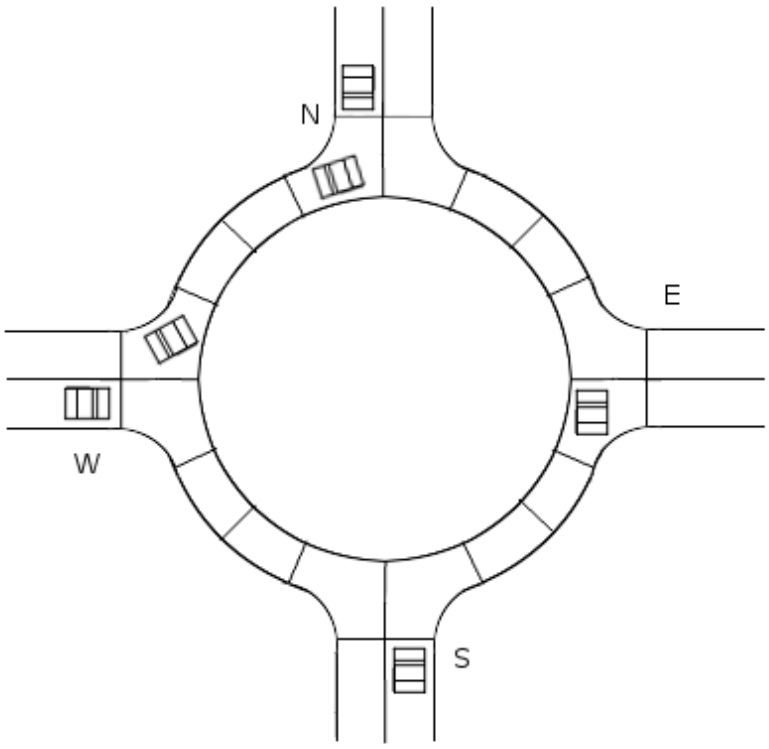
Zastosowany algorym został oparty na podejściu zaprezentowanym w książce: Michael J. Quinn, "Parallel Programing in C with MPI and OpenMPI".

Implementacja została zrealizowana w języku C++, przy wykorzystaniu standardu MPI w wersji 3.1, a także biblioteki generatorów liczb pseudolosowych SPRNG w wersji 5. Sama aplikacja została napisana w taki sposób, aby możliwe było jej równoległe uruchomienie na dowolnie wybranej liczbie węzłów.

2 Działanie algorytmu

Dla celów symulacji zakładamy tradycyjne założenia dotyczące ruchu na rondzie: samochody poruszają się w kierunku przeciwnym do ruchu wskazówek zegara, natomiast pojazdy znajdujące się obecnie na rondzie mają pierwszeństwo przed pojazdami wjeżdżającymi na nie.

Główny obszar reprezentujący rondo, podzielony jest na 16 segmentów (podczas pojedynczej iteracji wszystkie auta przemieszczają się o jedną komórkę), występują także 4 możliwe zjazdy/wjazdy na rondo (oznaczone jako N,S,W,E). Całość przedstawia poglądowo poniższy schemat:



Pojazd który chce wjechać na rondo, może dokonać tego manewru tylko i wyłącznie wtedy, gdy żaden inny samochód poruszający się obecnie po rondzie nie próbuje przemieścić się do tej samej komórki.

Do symulacji wykorzystane są 2 podstawowe struktury danych:
- tablica f, zawierająca średnie odstępy czasowe pomiędzy nadjeżdżającymi pojazdami dla każdego z 4 wjazdów
- macierz d o rozmiarze 4x4, w której poszczególne komórki przedstawiają prawdopodobieństwo, iż samochód wjeżdżający na rondo wjazdem „i" opuści go zjazdem „j"

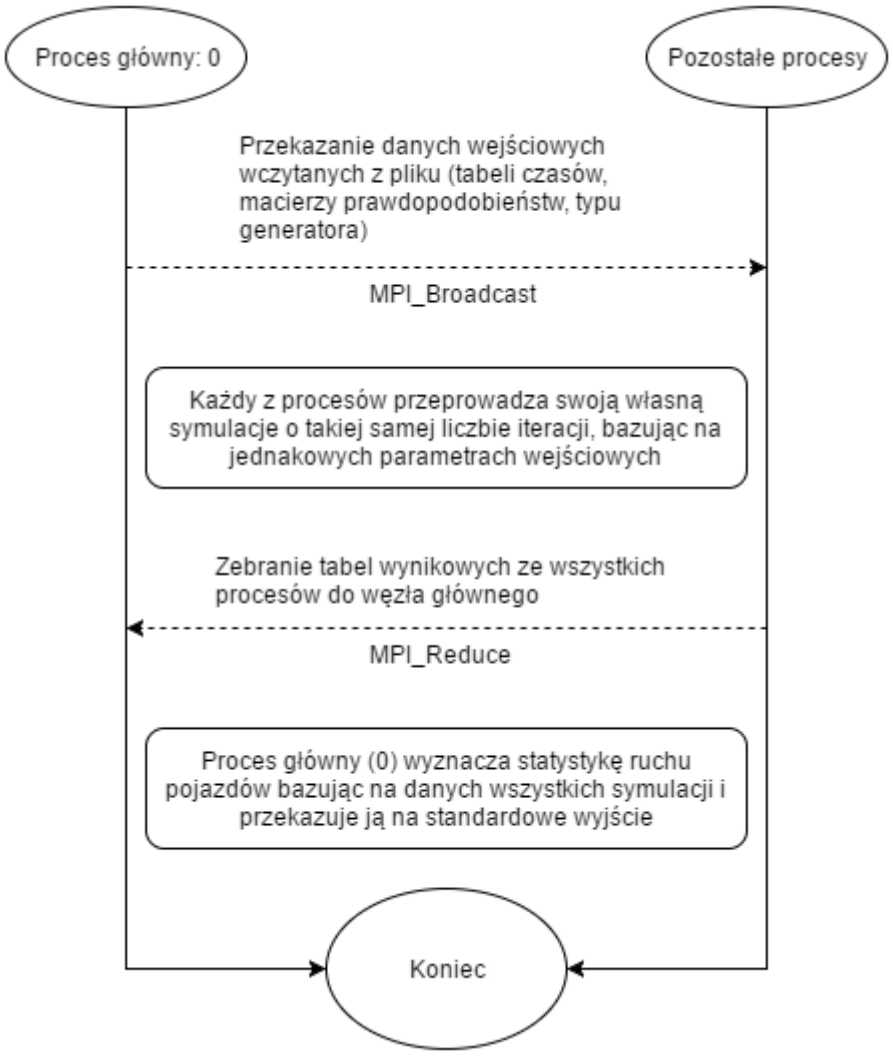
Przebieg każdej iteracji programu może zostać podzielony na 3 charakterystyczne kroki:

- dotarcie nowych pojazdów do ronda
- samochody znajdujące się już na rondzie przemieszczają się do kolejnej komórki. Pojazdy, które docierają do swojego zjazdu opuszczają skrzyżowanie, a ich ruch nie jest już dalej symulowany.
- Po przemieszczeniu samochodów algorytm sprawdza, czy komórka odpowiadająca danemu wjazdowi jest pusta. Jeśli tak, na rondo wpuszczane są pojazdy oczekujące w kolejce, bądź w razie pustej kolejki, te które dopiero docierają do skrzyżowania. Jeśli wjazd na rondo nie jest możliwy, samochody dodawane są do kolejki oczekujących.

Rezultatem programu będzie statystyka dla każdego z 4 wjazdów/zjazdów, udzielająca odpowiedzi na 2 zasadnicze pytania:

- jakie jest prawdopodobiestwo sytuacji, iż samochód docierający do ronda będzie musiał oczekiwać na wjazd
- jaka jest średnia długość kolejki oczekiwujących pojazdów

Rozproszenie w naszym programie zostało zrealizowane w następujący sposób:
na każdej z n maszyn uruchamiana jest osobna symulacja ruchu o takiej samej liczbie iteracji. Na końcu działania programu, wszystkie rezultaty są gromadzone przez proces główny w celu wyciągnięcia wniosków. Takie działanie może zostać zilustrowane następującym schematem:



3 Format danych wejściowych

Wspomniane wcześniej tablice średnich czasów i prawdopodobieństw (f i d) oraz typ generatora liczb pseudolosowych SPRNG, wczytywane są do programu z pliku wejściowego. Aby działanie aplikacji było poprawne należy upewnić się iż poniższy format danych jest zachowany:

- wiersz pierwszy zawiera pojedynczy numer generatora: liczba z zakresu 0-5
- wiersz drugi zawiera tablicę średnich czasów f, gdzie poszczególne komórki oddzielone są spacjami. Tablica składa się dokładnie z 4 elementów, po jednym dla każdego z wjazdów
- wiersze 3-6 zawierają macierz o wymiarze 4x4, gdzie poszczególne komórki wiersza oddzielone są spacjami

Przykładowy format danych wejściowych:

```
2
3 3 4 2
0.1 0.2 0.5 0.2
0.2 0.1 0.3 0.4
0.5 0.1 0.1 0.3
0.3 0.4 0.2 0.1
```

4 Obsługa aplikacji

Do obsługi programu przygotowany został plik Makefile oferujący następujące komendy:

- make - kompiluje program
- make run - tworzy plik nodes zawierający listę maszyn pracowni komputerowej na których będzie działać program, a następnie uruchamia na nich 16 instancji
- make local - uruchamia 16 instancji programu bazując na pojedynczej maszynie
- make clean - przywraca katalog do stanu wyjściowego

Należy mieć na uwadzę fakt, iż plik Makefile został przygotowany w taki sposób, by możliwe było jego uruchomienie na zajęciach. W przypadku używania programu na innej maszynie, należy odpowiednio zmodyfikować ścieżki dostępu do kompilatora mpicxx oraz bibliotek generatora SPRNG, zależnie od miejsca ich instalacji.