Temat 6: reakcja Biełousowa-Żabotyńskiego

Jakub Sawicki

19 października 2015

1 Reakcja Biełousowa-Żabotyńskiego

Jest to automat komórkowy, w którym komórki mogą znajdować się w następujących stanach:

- *Stan zdrowy — 0*: nowy stan to $a/k_1 + b/k_2$, gdzie a to ilość zarażonych komórek wśród 8 sąsiadów a b to ilość chorych sąsiednich komórek.
- *Stan chory — q*: w następnym kroku zdrowieje, tzn. jej stan to 0.
- *Stan zarażony — 1 do q − 1*: nowy stan to $s/(a+b+1)+g$, gdzie s jest sumą stanów komórki i sąsiadów, pozostałe oznaczenia jak dla komórki zdrowej.

Wartości k_1, k_2, q oraz g są stałe. Występujące operacje dzielenia wykonywane są w przestrzeni liczb całkowitych.

Na potrzeby symulacji ustawione zostały wartości $k_1 = 2$, $k_2 = 3$, $q = 10$ oraz $g = 3$. [2]

2 Analiza PCAM

Analiza rozbita została na bloki: partition, communication, agglomeration oraz mapping. [1]

2.1 Podział

Komórki automatu umieszczone są w dwuwymiarowej siatce, z czego każda komórka pamięta swój stan i jest w stanie przejść do następnego stanu bazując na stanie komórek sąsiednich. Podstawowym zadaniem będzie więc taka komórka.

Zakładając, że siatka jest kwadratowa mamy n^2 takich zadań. Każde zadanie przechowuje dane w postaci jednej liczby naturalnej O(1) i ma stały czas wykonania O(1) (przy zaniedbaniu komunikacji).

2.2 Komunikacja

Wymiana komunikatów następować musi pomiędzy sąsiednimi komórkami. Każda komórka musi dowiedzieć się o stanie swoich 8 sąsiadów.

Daje nam to w sumie $8n^2$ komunikatów o wielkości 1 (umownie) do wysłania na krok czasowy.

2.3 Aglomeracja

Optymalnym pod względem redukcji komunikacji jest podział kolumnowy.

```
1122334455
1122334455
1122334455
...
1122334455
```

Komunikacja zachodzi w tym przypadku jedynie pomiędzy sąsiednimi kolumnami. Daje to redukcję w ilości komunikatów do $2t$, gdzie t jest ilością bloków. Każdy komunikat jest też w tym przypadku większy, ma rozmiar n.

2.4 Mapowanie

W zależności od tego ile jest dostępnych procesorów/wątków, na tyle bloków można podzielić siatkę. Dzięki temu obliczenia dla poszczególnych bloków nie będą sobie wchodzić w drogę. Warto też zadbać o lokalność komunikacji umieszczając sąsiednie bloki na fizycznie bliskich procesorach.

3 Implementacja

Kod programu znajduje się pod github.com/jswk/AR/lab1/CA.c. Został on napisany w C z wykorzystaniem MPI do komunikacji.

Ustawienie stałej DEBUG pozwala wypisywać stan symulacji na standardowe wyjście. Na podstawie pliku wyjścia za pomocą skryptu process.sh możliwe jest wygenerowanie wizualizacji procesu.

Struktury wygenerowane przez program widoczne są na Rys. 1.

4 Wydajność

Program uruchomiony został na klastrze Zeus. Pomiary obejmowały różne ilości aktywnych procesorów, od 1 do 24 (dwa node'y po 12 rdzeni).

Przeprowadzone zostało kilka pomiarów. Pierwszy dla wielkości problemu $n = 200$. Następnie dla $n = 1000$ oraz dla skalowanego problemu (ok. 10^6 komórek na procesor).

Wyniki dla $n = 200$ przedstawione zostały na Rys. 2. Problem ten był zbyt mały dla ilości procesorów większej niż 5–6. Dla większych ilości procesorów narzut komunikacyjny prowadził do zmniejszenia efektywności.

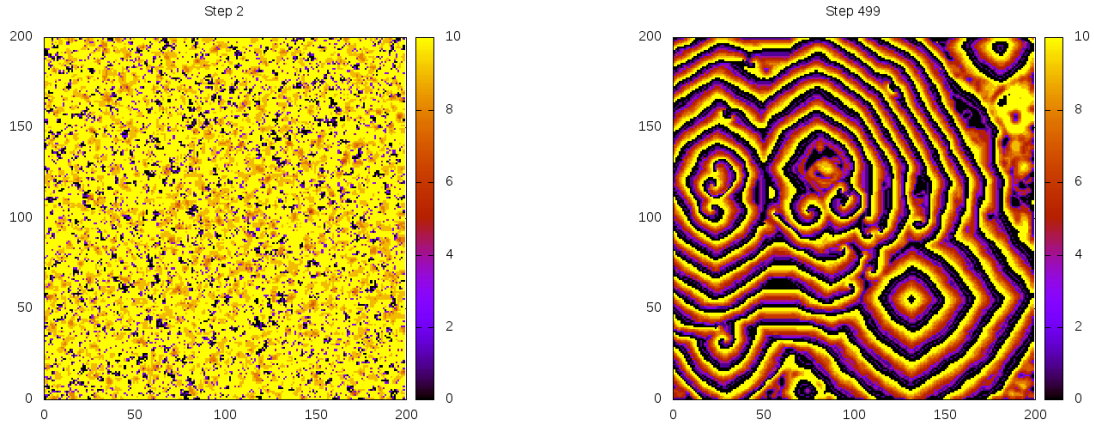
Dla $n = 1000$ wyniki przedstawione są na Rys. 3. Tutaj większa ilość procesorów daje większy wzrost wydajności, efektywność nie opada tak szybko jak w przypadku poprzednim.

Dla problemu skalowanego wyniki są na Rys. 4. Rozmiar problemu skalowany był w taki sposób, aby zawsze na jeden procesor przypadało ok. 10^6 komórek.

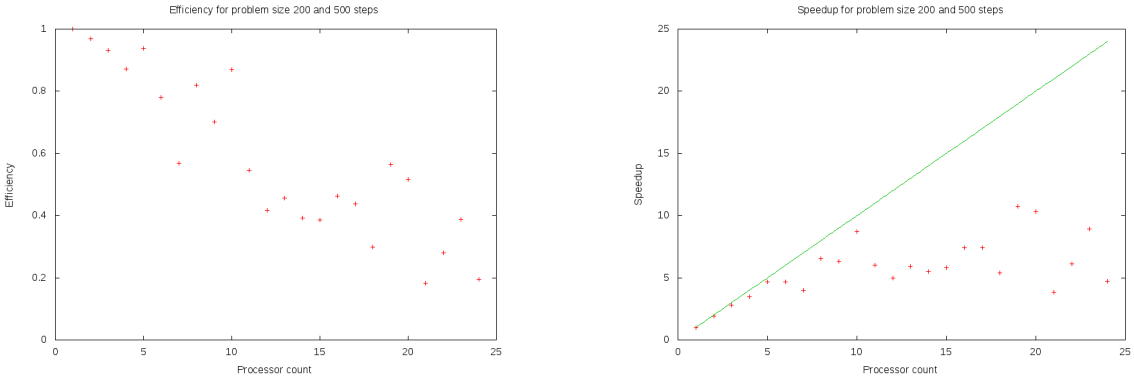
Literatura

[1] I. Foster: *Designing and Building Parallel Programs* http://www.mcs.anl.gov/dbpp/ dostęp 2015.10.14

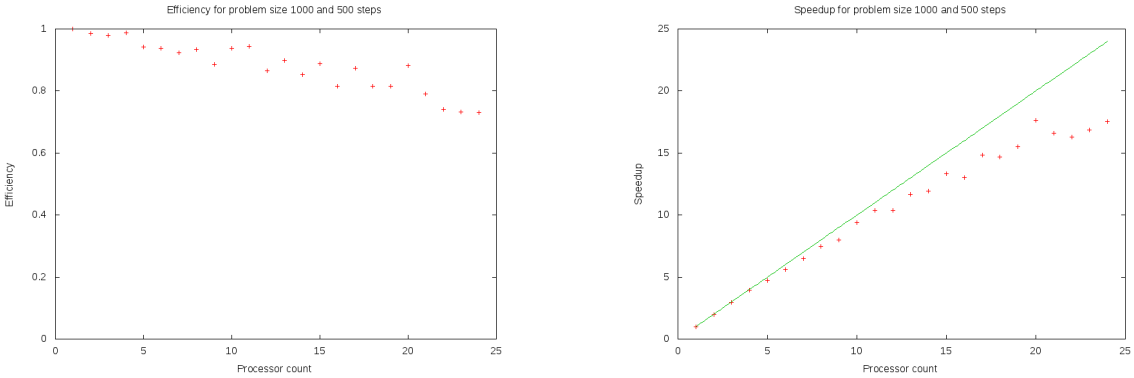
[2] Hermetic Systems: *Five Cellular Automata: The Belousov-Zhabotinsky Reaction* http://www.hermetic.ch/pca/bz.htm dostęp 2015.03.04



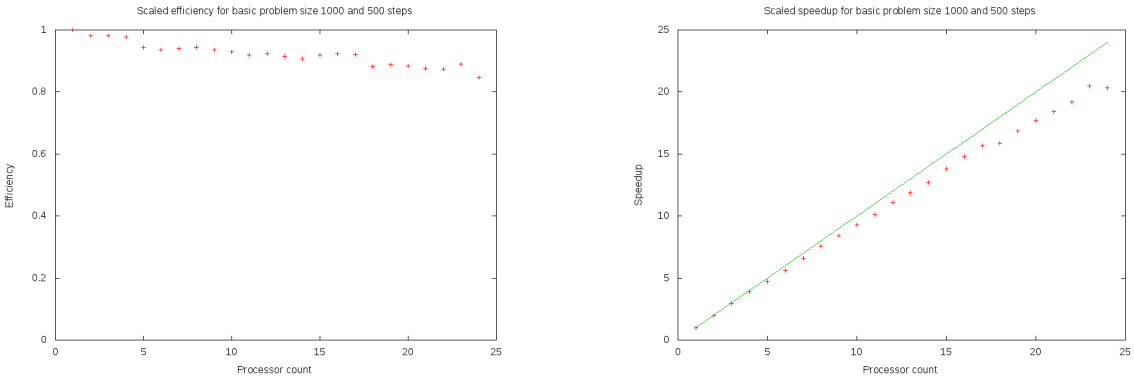
Rys. 1: Wykresy pokazują stan automatu dla 2. i 499. iteracji. Kolorem oznaczony jest stan poszczególnych komórek. Zobaczyć można wykształcenie zorganizowanych struktur z początkowo losowego stanu.



Rys. 2: Wykresy pokazują efektywność oraz przyspieszenie dla problemu wielkości $n = 200$.



Rys. 3: Wykresy pokazują efektywność oraz przyspieszenie dla problemu wielkości $n = 1000$.



Rys. 4: Wykresy pokazują efektywność oraz przyspieszenie dla skalowanego problemu, ok. 10^6 komórek na procesor.