PROJEKT PORR

Katarzyna Kucharczyk
Julian Maciejewski

1. Treść zadania - nr 24

Porównanie efektywności różnych metod wektoryzacji i zrównoleglenia algorytmu programowania dynamicznego w zadaniu optymalnej syntezy na horyzoncie nieskończonym z dyskontem.

Zadanie jest opracowane na podstawie zadania zadanie sterowania systemem wielozbiornikowym Górnej Wisły.

2. Algorytm

Algorytmem zrównelaglanym jest zadanie optymalnej syntezy polityk przy użyciu dwóch metod minmaksowej oraz z dyskontem. Implementacja powstawła na podstawie artykułu "Assessment of the Cell Broadband Engine Architecture as a platform to solve closed-loop optimal control problems" autorstwa Andrzeja Karbowskiego oraz Macieja Ramiszewskiego. Na podstawie artykułu ponadto Dane zostały pobrane z dostarczonych plików zawierających historyczne pomiary rezerwuarów zbiorników wodnych Wisły.

Pseudokod:

```
while r > epsilon
  foreach step k do
       foreach state resevoir level x (Sg, St) do
       best_Q = 999999999.9
       foreach control u do
    Q=0;
    foreach influence w
      switch aproach do
       case minmax:
       g_J = max(g(x,u,w),Jprev(k,x,u,i),knext)
       Q = max(Q,g_J)
       break
       case dicount:
       g_J = g(x,u,i) + alpha*Jprev(k,x,u,i)
       Q = Q + pw*g_J
              break
              end
      end
    end
       if Q < best_Q do
      best_Q = Q
```

```
            bestControl = u
          end
             end                    }
             Policy[k,x] = bestControl
             J[k,x] = bestQ
             end
        end
        r = || J - Jprev ||
        Jprev = J
      end
```

---

3. Testy

a) środowisko testowe:

Wszystkie testy zostały wykonane dla takich samych warunków początkowych. Dla konfiguracji sterowań było to:

- rG1division = 7;
- rG2division = 5;
- rT1division = 7;
- rT2division = 5;
- rT3division = 6;
- r3division = 7;

Gdzie im większa liczba przy każdym z parametrów tym na więcej testowych elementów podzielony jest przedział sterowań i tym więcej kombinacji powstawało do przetestowania.
Epsilon dla głównej pętli był o wartości 0.01.
Ilość kroków była równa 36.
Przedziały stanów zostały podzielone na 7 przedziałów, co dawało 49 kombinacji różnych stanów.

b) profilowanie

Zostało sprawdzone profilowanie za pomocą programu prof dla obu metod.

- minmax

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self   total
 time   seconds   seconds  calls  ms/call  ms/call  name
 30.30       35.18  35.18 544546800     0.00   0.00    koszt
 20.85       59.39  24.21 4356374400    0.00   0.00   maxy
 18.40       80.77  21.37 544546800     0.00   0.00   g
 12.43       95.21  14.44                             main
 10.37 107.25 12.04 544546800     0.00   0.00          J
  6.41 114.69       7.45 261383444      0.00   0.00   convertNewFloatStateTo
```

```
  0.63 115.43       0.73         3   243.64   243.64  wczytajHist
  0.47 115.97       0.54                              readMatrix
  0.19 116.19       0.22   3528   0.06   0.06  addJ
  0.00 116.19       0.00         2      0.00   0.00   norm
  0.00 116.19       0.00         1      0.00   0.00   generateControls
  0.00 116.19       0.00         1      0.00   0.00   generateInflows
  0.00 116.19       0.00         1      0.00   0.00   generateStartStates
  0.00 116.19       0.00         1      0.00   0.00   prepareJ
  0.00 116.19       0.00         1      0.00   0.00   preparePolicy


[1]    99.5   14.44  101.21                     main [1]
              21.37   53.34 544546800/544546800 g [2]
              12.04  7.45 544546800/544546800   J [5]
              6.05   0.00 1089093600/4356374400 maxy [4]
              0.73   0.00         3/3           wczytajHist [7]
              0.22   0.00   3528/3528           addJ [9]
              0.00   0.00   3528/261383444      convertNewFloatStateTo [6]
              0.00   0.00         2/2           norm [10]
              0.00   0.00         1/1           preparePolicy [15]
              0.00   0.00         1/1           prepareJ [14]
              0.00   0.00         1/1           generateStartStates [13]
              0.00   0.00         1/1           generateControls [11]
              0.00   0.00         1/1           generateInflows [12]
-----------------------------------------------
              21.37   53.34 544546800/544546800 main [1]
[2]    64.3   21.37   53.34 544546800           g [2]
              35.18   18.16 544546800/544546800 koszt [3]
-----------------------------------------------
              35.18   18.16 544546800/544546800 g [2]
[3]    45.9   35.18   18.16 544546800           koszt [3]
              18.16  0.00 3267280800/4356374400 maxy [4]
-----------------------------------------------
              6.05   0.00 1089093600/4356374400 main [1]
              18.16  0.00 3267280800/4356374400 koszt [3]
[4]    20.8   24.21  0.00 4356374400     maxy [4]
-----------------------------------------------
              12.04  7.45 544546800/544546800   main [1]
[5]    16.8   12.04  7.45 544546800      J [5]
              7.45   0.00 261379916/261383444   convertNewFloatStateTo [6]
-----------------------------------------------
              0.00   0.00   3528/261383444      main [1]
              7.45   0.00 261379916/261383444   J [5]
[6]    6.4    7.45   0.00 261383444             convertNewFloatStateTo [6]
-----------------------------------------------
              0.73   0.00         3/3           main [1]
[7]    0.6    0.73   0.00         3             wczytajHist [7]
```

- z dyskontem

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 31.04       33.82  33.82 544546800     0.00     0.00  koszt
 18.36       53.83  20.01 544546800     0.00     0.00  g
 15.80       71.04  17.21 3267280800    0.00     0.00  maxy
 15.35       87.76  16.72                              main
 11.09       99.84  12.08 544546800     0.00     0.00  J
  7.29 107.78       7.94 261383444      0.00     0.00  convertNewFloatStateTo
  0.56 108.39       0.61          3   203.59   203.59  wczytajHist
  0.42 108.85       0.46                               readMatrix
  0.18 109.05       0.20   3528   0.06     0.06  addJ
  0.00 109.05       0.00          2      0.00     0.00  norm
  0.00 109.05       0.00          1      0.00     0.00  generateControls
  0.00 109.05       0.00          1      0.00     0.00  generateInflows
  0.00 109.05       0.00          1      0.00     0.00  generateStartStates
  0.00 109.05       0.00          1      0.00     0.00  prepareJ
  0.00 109.05       0.00          1      0.00     0.00  preparePolicy

granularity: each sample hit covers 2 byte(s) for 0.01% of 109.05 seconds

index % time  self  children      called       name
                                               <spontaneous>
[1]    99.6   16.72   91.87                    main [1]
              20.01   51.03 544546800/544546800 g [2]
              12.08  7.94 544546800/544546800   J [4]
              0.61   0.00          3/3          wczytajHist [7]
              0.20   0.00   3528/3528           addJ [9]
              0.00   0.00   3528/261383444      convertNewFloatStateTo [6]
              0.00   0.00          2/2          norm [10]
              0.00   0.00          1/1          preparePolicy [15]
              0.00   0.00          1/1          prepareJ [14]
              0.00   0.00          1/1          generateStartStates [13]
              0.00   0.00          1/1          generateControls [11]
              0.00   0.00          1/1          generateInflows [12]
-----------------------------------------------
              20.01   51.03 544546800/544546800 main [1]
[2]    65.1   20.01   51.03 544546800           g [2]
              33.82   17.21 544546800/544546800 koszt [3]
-----------------------------------------------
              33.82   17.21 544546800/544546800 g [2]
[3]    46.8   33.82   17.21 544546800           koszt [3]
              17.21  0.00 3267280800/3267280800 maxy [5]
-----------------------------------------------
```

```
             12.08  7.94 544546800/544546800   main [1]
[4]    18.4   12.08  7.94 544546800      J [4]
             7.94   0.00 261379916/261383444   convertNewFloatStateTo [6]
-----------------------------------------------
             17.21  0.00 3267280800/3267280800 koszt [3]
[5]    15.8   17.21  0.00 3267280800           maxy [5]
-----------------------------------------------
             0.00   0.00   3528/261383444      main [1]
             7.94   0.00 261379916/261383444   J [4]
[6]    7.3    7.94   0.00 261383444            convertNewFloatStateTo [6]
```

W obu przypadkach najbardziej kosztownym jest wyliczanie J oraz kosztu.

c) wektoryzacja automatyczna

	minmax	**discount**
brak wektoryzacji	**real 3m6.201s** **user 3m5.252s** **sys 0m0.276s**	**real 2m50.474s** **user 2m49.032s** **sys 0m0.488s**
flaga -O2	**real 0m52.374s** **user 0m52.012s** **sys 0m0.180s** współczynnik 3.25	**real 0m50.629s** **user 0m50.344s** **sys 0m0.068s** współczynnik 2.94
flaga -O2 -ftree-vectorize	**real 0m51.631s** **user 0m51.216s** **sys 0m0.256s** współczynnik 3.078	**real 0m50.945s** **user 0m50.556s** **sys 0m0.152s** współczynnik 2.70

Metoda z dyskontem przy tych samych warunkach początkowych szybciej kończy swoją pracę.

Można zauważyć, że wektoryzacja przy użyciu flagi -O2 zmniejszyła czas wykonywania prawie trzykrotnie. Przy zastosowaniu kombinacji -O2 -ftree-vectorize, czas już niewiele bardziej się zmniejszył. Jednak można zauważyć różnicę, że przy tej fladze większa czas wykonywania jest na poziomie jądra systemu (kernel mode) niż użytkownika (user mode).

d) użycie flagi "-msse"

	minmax	**discount**
brak	**real 2m3.742s**	**real 1m46.132s**

wektoryzacji "-msse"	**user 2m2.344s** **sys 0m0.208s**	**user 1m42.808s** **sys 0m0.484s**
flaga -O2 "-msse"	**real 0m52.374s** **user 0m52.012s** **sys 0m0.180s** współczynnik 2.32	**real 0m30.693s** **user 0m30.412s** **sys 0m0.004s** współczynnik 2.93
flaga -O2 -ftree-vectorize "-msse"	**real 0m30.005s** **user 0m29.936s** **sys 0m0.000s** współczynnik4.50	**real 0m31.302s** **user 0m30.808s** **sys 0m0.108s** współczynnik 3.00

Przy użyciu flagi -msse zauważalna była już znaczna poprawa w porównaniu z pierwszą kopilacją (w przypadku minmax czas zmniejszyl sie o ⅓ a w przypadku metody discount o połowę).W przypadku flagi -O2 czasy nie zmmieniły się, jednak przy dodanej dlasze -ftree-wektorize czas wykonania testów spadł prawie 4.5-krotnie.

e) Wektoryzacja manualna

Jak się okazuje do tak skomplikowanego programu bardzo jest ciężko dobrać właściwą metodę do werktoryzacji manualnej. Została wybrana funkcja normalizacji. Funkcja ta zgodnie z danych z programu prof, nie jest zbyt kosztowną operacją tak więc zysk z przerabiania funkcji nie okazał się być miarodajny w porownaniu wektoryzajami manualnymi. Większa ilość testów oznaczałaby całkowitą refaktoryzacje kodu.

f) OpenMP

Poprawę dzięki użyciu programu OpenMP można zauważyć w kolejnych etapach:

- równoleglenie przepisywania macierzy Jprev jako J:
real 1m1.789s
user 1m1.376s
sys 0m0.000s

- dodanie dyrekrywy przy normalizacji macierzy
real 1m1.430s
user 1m1.020s
sys 0m0.004s

- dodanie przy generowaniu stanow i wczytywaniu z plikow
real 1m1.154s
user 1m0.932s

sys 0m0.008s

Można zauważyć największy skok w zmianie czasu wykonania przy przepisywaniu macierzy. Kolejne akcje miały wpływ lecz już niewielki.

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 46.95       50.31  50.31 544546802     0.00     0.00  wczytajHist
 19.28       70.97  20.66 1089097129    0.00     0.00  prepareJ
 15.42       87.49  16.52         1     16.52  87.49   main
  5.13       92.99  5.50                                preparePolicy
  4.42       97.73  4.74          1      4.74   4.74   J
  4.10 102.12       4.40                                addJ
  3.84 106.23       4.11 131019238       0.00   0.00   g
  0.66 106.94       0.71                                frame_dummy
  0.29 107.26       0.32                                __do_global_dtors_aux
  0.00 107.26       0.00          3      0.00   0.00   generateControls
  0.00 107.26       0.00          1      0.00   0.00   convertNewFloatStateTo
  0.00 107.26       0.00          1      0.00   0.00   generateInflows
```

Ponadto zmienił się procentowo czas najbardziej obciązonych operacji.

4. Wnioski

Okazuje się, że wektoryzacja ręczna pozwala nawet na kilkukrotne przyspieszenie programu, jednak nie każdą funkcję można efektywnie wektoryzować. Czasami specyficzne sformułowanie funkcji w dużej mierze uniemożliwia wektoryzację i wymaga pracochłonnej przebudowy funkcji. Jak się okazuje równie dobre efekty daje zrównoleglenie kodu. Jednak ważną rzeczą jest zwrócenie uwagi czy dany fragment kodu nadaje się do zrównoleglenia, gdyż czasem złe użyte dyrektywy mogą prowadzić do wydłużenia czasu działania programu lub brak efektu.