ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996

Seria: INFORMATYKA z. 31 Nr kol. 1348

Marcin SKOWRONEK

RÓWNOLEGŁA REALIZACJA EKSPERYMENTÓW Z WYKORZYSTANIEM PAKIETU DarP

Streszczenie. W pracy opisano zasady koordynacji oraz przedstawiono przykładowe algorytmy synchronizacji i komunikacji współpracujących ze sobą procesów równoległych. Opisano dodatkowe elementy pakietu DarP i zilustrowano ich użycie na przykładzie budowy modeli procesu *Nadawcy* i procesu *Wykonawcy*. Zamieszczone wyniki badań ilustrują osiągnięte przyśpieszenia.

PARALLEL EXPERIMENTS EXECUTION USING DarP PACKAGE

Summary. In the paper were presented the rules of coordination as well as some as examples of synchronization and communication algorithms for the parallel cooperating processes. Some additional elements of DarP package were also described and their usage was illustrated on the example of *Sender* and *Executor* processes models. The results of modelling tasks illustrate the speed up that was achieved.

PARALLELE AUSFÜHRUNG DER EXPERIMENTE MIT DER AUSNUTZUNG DES DarP-PAKETS

Zusammenfassung. Im Artikel wurden Prinzipien der Koordination sowie Musteralgorithmen der Synchronisation und Kommunikation von den parallelen Prozessen, die miteinander zusammenarbeiten, dargestellt. Auch zusätzliche Elemente des DarP-Pakets hat man beschrieben. Ihre Anwendung wurde am Beispiel des Aufbaus von Modellen der Prozesse *Der Absender* und *Der Ausführer* gezeigt. Die im Artikel enthaltenen Versuchsergebnisse illustrieren erreichte Beschleunigungen.

1. Wprowadzenie

Badania modelowe sprowadzają się zazwyczaj do wykonania sekwencji eksperymentów z wykorzystaniem modelu badanego układu dynamicznego. W przypadku modelowania układów ciągłych *eksperymentem* jest rozwiązanie zagadnienia początkowego, to jest rozwiązanie układu równań różniczkowych w ustalonym przedziale czasu, przy zadanych warunkach początkowych i określonych parametrach modelu. Wynikiem eksperymentu są końcowe wartości lub ciągi liczbowe prezentujące przebiegi zmiennych opisu modelu.

Przykładem narzędzia programowego umożliwiającego budowę modeli układów ciągłych i realizację badań modelowych jest pakiet DarP [5]. Utworzone za pomocą pakietu modele umożliwiają zarówno wsadową, jak i interakcyjną realizację badań modelowych. Do interakcyjnej obsługi badań modelowych udostępniany jest rozbudowany system menu i system podpowiedzi. Pakiet ten standardowo umożliwia realizację eksperymentów w jednej stacji systemu komputerowego. W ramach badań opisanych w pracach [3] i [4] modyfikowano ten pakiet tak, by umożliwiał on budowę modeli i równoległą realizację sekwencji eksperymentów w sieci NetWare.

Elementarnym zadaniem wykonywanym w trakcie badań modelowych jest realizacja eksperymentu, któremu w zapisie algorytmu badań modelowych, stosując konwencję zapisu przedstawioną w pracy [2], odpowiada instrukcja podstawienia

$$y_i = \mathrm{E}(a_i), \tag{1}$$

gdzie:

a_i – dane wejściowe operacji,

y_i – wynik operacji.

W zadaniach modelowania układów ciągłych danymi wejściowymi operacji E są warunki początkowe zmiennych stanu i/lub parametry modelu. Wynikiem operacji E są końcowe wartości lub ciągi liczbowe prezentujące przebiegi wybranych zmiennych opisu modelu. W tym drugim przypadku pośrednim wynikiem eksperymentu są nazwy plików, w których zapisane są ciągi rozwiązań.

Fragmenty algorytmu badań modelowych, opisujące sekwencje eksperymentów, które mogą być wykonywane równolegle, mają postać dwóch instrukcji:

1. Wyznacz ciąg danych $a_1, a_2, ..., a_M$.
2. Dla $i = 1, 2, ..., M$ wyznacz wyniki $y_i = \mathrm{E}(a_i)$.

Instrukcja 1 spełnia tu rolę instrukcji inicjującej (embrionu), natomiast instrukcja 2 jest zapisem sekwencji eksperymentów, których realizacja może być wykonywana równolegle w systemie rozproszonym. Elementarne przykłady takich sekwencji to: wyznaczanie sklado-

wych gradientu [4], wyznaczanie zmienności wskaźnika jakości w zadanym obszarze parametrów wejściowych [3], znajdowanie macierzy tranzycji w liniowych zadaniach dwugranicznych, rozwiązywanie zagadnień brzegowych metodami Monte Carlo.

2. Koordynacja równoległej realizacji eksperymentów

Wprowadzone w pakiecie DarP mechanizmy, umożliwiające budowę modeli i równoległą realizację eksperymentów, wykorzystują do synchronizacji i komunikacji między procesami pliki o ustalonych nazwach umieszczone na wspólnym dysku serwera sieci. Szczegółowe zasady realizacji takiej komunikacji opisano w pracach [1] i [3].

Przyjmuje się, że zadanie wykonania sekwencji M eksperymentów sterowane jest przez proces zwany *Nadawcą*. Eksperymenty wykonywane są przez procesy zwane *Wykonawcami*, których liczba N jest znana *Nadawcy*. Proces *Nadawcy* i procesy *Wykonawców* realizowane są w różnych stacjach sieci komputerowej. Przed rozpoczęciem sekwencji eksperymentów są uruchomione procesy *Wykonawców* i oczekują one na komunikat od *Nadawcy*. Każdy z procesów *Wykonawców* zainicjowany jest z innego katalogu bieżącego. Przyjęcie takiego założenia wynika stąd, że modele zbudowane za pomocą dostępnych środków modelowania często korzystają z plików roboczych o ustalonych nazwach. Inicjowanie procesów *Wykonawców* w oddzielnych katalogach jest tu prostym sposobem uniknięcia kolizji dostępu do plików roboczych modelu. Przyjęto, że procesy *Wykonawców* inicjowane są w katalogach o nazwach W1, W2, ..., WN podrzędnych w stosunku do katalogu, w którym zainicjowano proces *Nadawcy*.

Elementarne operacje wykonywane w procesie *Nadawcy* podczas zlecania wykonania pojedynczego eksperymentu to:

1. Dla każdego eksperymentu w sekwencji:
   - przygotuj plik z danymi,
   - wywołaj procedurę wyślij_odczytaj_komunikat.
2. Dopóki nie odczytano wszystkich wyników:
   - wywołaj procedurę wyślij_odczytaj_komunikat.

Procedura wyślij_odczytaj_komunikat składa się z dwóch faz:

a) fazy sprawdzania, czy istnieje wolny *Wykonawca* i wysłania komunikatu do *Wykonawcy*,

b) fazy sprawdzania, czy istnieją komunikaty odpowiedzi od *Wykonawców* i ich odczytu.

Faza a) wykonywana jest tylko wtedy, gdy przygotowany jest plik z danymi i istnieje nie zajęty *Wykonawca*. Jeżeli jest przygotowany plik z danymi, to fazy a) i b) wykonywane są na

przemian aż do jednorazowego wykonania fazy a). Jeżeli nie jest przygotowany plik z danymi, to wykonywana jest tylko faza b).

Operacje wykonywane w fazach a) i b) są następujące:

a) jeżeli istnieje wolny *Wykonawca*, to:
   1) przemianuj plik z danymi na komunikat do określonego *Wykonawcy*,
   2) zaznacz wysłanie komunikatu,
   3) zaznacz zajęcie *Wykonawcy*,

b) dopóki istnieją komunikaty od *Wykonawców*, to:
   1) przemianuj plik odpowiedzi,
   2) zaznacz zwolnienie *Wykonawcy*,
   3) zaznacz odebranie odpowiedzi,
   4) odczytaj wyniki eksperymentu,
   5) usuń plik wyników.

Przyjęto tu rozwiązanie, że komunikaty do *Wykonawców* kierowane są do ich katalogów bieżących, natomiast *Wykonawcy* kierują odpowiedzi do katalogu, z którego uruchomiono proces *Nadawcy*, przy czym nazwa pliku stanowiącego komunikat odpowiedzi wskazuje *Wykonawcę* (rys. 1). Po wysłaniu wszystkich zleceń wykonania eksperymentów wykonuje się cyklicznie fazę d), aż do momentu odebrania wszystkich odpowiedzi. Poprawne wykonanie operacji przemianowania pliku wymaga, by nie istniał plik docelowy.

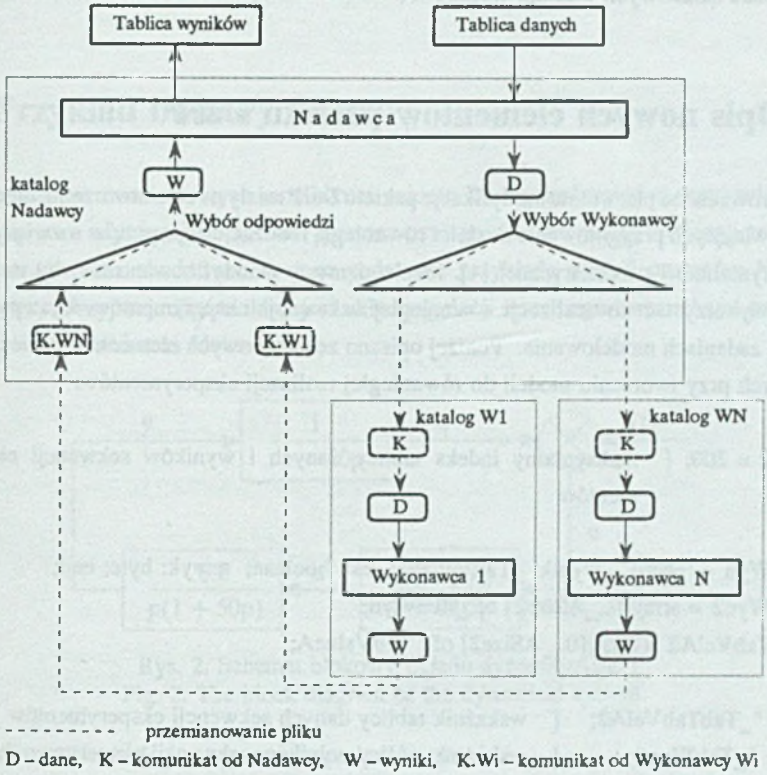
Operacje wykonywane w procesie *Wykonawcy* to:

a) czekaj na komunikat od *Nadawcy*,
b) przemianuj plik komunikatu od *Nadawcy*,
c) odczytaj dane,
d) wykonaj eksperyment,
e) zapisz wyniki eksperymentu do pliku wyników,
f) przemianuj plik wyników na komunikat do *Nadawcy*,
g) skocz do a).

Analizując operacje wykonywane przez *Nadawcę*, widzimy, że są one niezależne od sposobu realizacji eksperymentu. Dla użytkownika natomiast nieistotne są szczegóły związane ze sposobem realizacji komunikacji. Wykonaniu sekwencji eksperymentów może więc w programie procesu *Nadawcy* odpowiadać wywołanie jednej procedury, która odczytuje dane o parametrach poszczególnych eksperymentów (a_1, a_2, ..., a_M) z tablicy danych i zapisuje wyniki eksperymentów (y_1, y_2, ..., y_M) do tablicy wyników. Parametrami sterującymi procedury wykonania sekwencji eksperymentów są:

- liczba zlecanych eksperymentów M,
- indeks tablicy danych, od którego rozpoczynają się dane sekwencji eksperymentów,
- liczba składowych wektora danych wejściowych,

- liczba wykonawców *N*,
- nazwy lub wskaźniki tablic danych i wyników.



Rys. 1. Komunikacja między *Nadawcą* i *Wykonawcami*
Fig. 1. Communication between the *Sender* and *Executors*

Analizując operacje wykonywane w procesie *Wykonawcy*, widzimy, że jedyną operacją istotnie zależną od modelowanego zadania to operacja d). Przyjmując, że odczytane dane wejściowe eksperymentu i wyznaczone wyniki eksperymentu będą zapisane w ustalonych tablicach, można przyjąć, że w zapisie algorytmu procesu *Wykonawcy* operacji d) odpowiada wywołanie ustalonej procedury, której definicję przygotuje użytkownik. Przy takich założeniach program procesu *Wykonawcy*, oprócz typowych składowych opisujących równania modelu, powinien zawierać definicję procedury odpowiadającej realizacji operacji d), a w części wykonawczej programu dwie instrukcje:

1) przypisania nazwy procedury realizującej operację d) określonej zmiennej systemowej,
2) wywołania systemowej procedury realizującej algorytm procesu *Wykonawcy*.

Parametrami sterującymi tej procedury są:

- nazwy lub wskaźniki tablic danych i wyników pojedynczego eksperymentu,
- liczba składowych wektora wyników.

3. Opis nowych elementów pakietu DarP

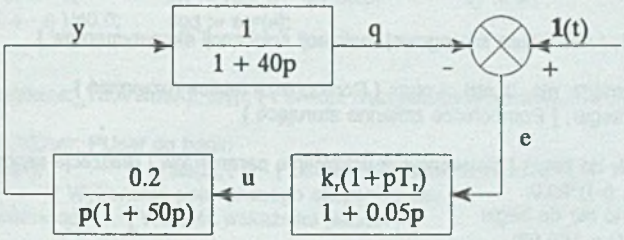
Przeprowadzone pierwotnie modyfikacje pakietu DarP miały na celu stworzenie mechanizmów ułatwiających przygotowanie modeli i równoległą realizację algorytmów rozwiązywania zadań optymalizacji parametrycznej [4]. Mechanizmy te zmodyfikowano tak, by można je również wykorzystać do realizacji równoległej sekwencji eksperymentów występujących w innych zadaniach modelowania. Poniżej opisano zestaw nowych elementów pakietu DarP użytecznych przy tworzeniu modeli do równoległej realizacji eksperymentów:

```
Const
  _ASize2 = 200; { maksymalny indeks tablicy danych i wyników sekwencji eksperymentów
Type
  _RecWyn = record   wynik: _TabValueA; jest: boolean;  nrwyk: byte; end;
  _TabWyn2 = array[0.._ASize2] of _RecWyn;
  _TabTabValA2 = array[0.._ASize2] of _TabValueA;
Var
  _ptd: ^_TabTabValA2;  { wskaźnik tablicy danych sekwencji eksperymentów }
  _ptw: ^_TabWyn2;      { wskaźnik tablicy wyników sekwencji eksperymentów }
  _wd:  _TabValueA;     { wektor danych pojedynczego eksperymentu }
  _ww:  _TabValueA;     { wektor wyników pojedynczego eksperymentu }
  RunFunc: procedure(da:_TabValueA; Var wa:_TabValueA);
        { Nazwa i nagłówek procedury wywoływanej w algorytmie procesu Wykonawcy,
          da – wektor danych wejściowych, wa – wektor wyników. }
Procedure Make(m0, m: integer; mwyk: byte);
  { Procedura zlecająca wykonanie sekwencji eksperymentów,
    m0    – indeks tablicy danych i wyników dla pierwszego eksperymentu,
    m     – indeks tablicy danych i wyników dla ostatniego eksperymentu,
    mwyk  – liczba wykonawców (mwyk<=0 – eksperymenty wykonywane są na
            stacji nadawcy. }
```

```
Procedure Execution(mm: integer);
  { Procedura realizująca algorytm wykonawcy,
    mm  –  liczba składowych wektora wyniku. }
```

4. Przykład budowy modeli

W celu zilustrowania nowych możliwości pakietu DarP przedstawiono przykład modelu opracowanego z użyciem pakietu DarP przeznaczonego do szeregowej realizacji sekwencji eksperymentów oraz zbudowane na jego podstawie przykłady modeli procesów *Nadawcy* i *Wykonawcy*. Przyjmuje się, że dany jest układ regulacji z zerowymi warunkami początkowymi, opisany schematem blokowym



Rys. 2. Schemat blokowy układu dynamicznego
Fig. 2. The block diagram of the dynamical system

Celem modelowania jest wyznaczenie charakterystyki

$$Q(k_r, T_r) = \int_0^{1000} e^2\,dt \tag{2}$$

dla k_r = 0.02, 0.04, ..., 0.32 (16 wartości) i T_r = 0, 20, ..., 100 (6 wartości). Wyniki sekwencji eksperymentów mają być umieszczone w pliku rozwiązań, jako 6 funkcji zmiennej k_r, opisanych nazwami 'q0', 'q20', ..., 'q100'.

Przykładowy opis modelu z szeregową realizacją sekwencji eksperymentów przedstawia poniższy wydruk:

```
program model1; { Szeregowa realizacja sekwencji eksperymentów (model szeregowy) }
{$M 16384,10000,40000}
uses Dos, Crt, DarVar, DarRun, DarKom, DarPDod, DarOpt;
Const
   nkr = 16;   ntr = 6; { Pomocnicze stałe określające rozmiar pomocniczej tablicy rozwiązań }
Var
   Xuser:    record  w,    y,    q,    x,    qq:    otyp;   end absolute _X;
```

```
    DXuser:     record dw,     dy,     dq,     dx,     dqq:    otyp;   end absolute _DX;
    YUser:      record s,      e,      u:      otyp;   end absolute _Y;
    Puser:      record kr,     tr:     otyp;   end absolute _P;
    t:          otyp absolute _T; { zmienna niezależna }
{$F+}
Procedure OpisUkladu; { Zapis równań stanu }
begin
    with XUser, DXUser, YUser, PUser do begin
        s := 1;                 e := - q + s;                   u := (tr*e + x)/0.05;
        dx := kr*e - u;         dw := (0.2*u - w)/ 50.0;        dy := w;
        dq := ( y - q )/40.0;   dqq := sqr(e);
    end;
end;
Function wskaznik(aa:_TabValueA): otyp; { Funkcja wyznaczająca wartość wskaźnika jakości }
begin
    with XUser, YUser, PUser do begin
        kr := aa[1];            tr := aa[2];  { Ustawianie parametrów modelu }
        Run;        { Wykonanie pojedynczego eksperymentu }
        wskaznik := qq;     { Wartość wskaźnika jakości }
    end;
end;
Procedure Akcja1; { Procedura szeregowej realizacji sekwencji eksperymentów }
var:
    tabq:       array[1..nkr, 0..ntr] of otyp; { Pomocnicza tablica rozwiązań }
    i, j:       integer; { Pomocnicze zmienne sterujące }
begin
    for j := 1 to ntr do begin { Sekwencyjne ustawianie parametrów i realizacja eksperymentów }
        _wd[2] := (j-1)*20.0;
        for i := 1 to nkr do begin
            _wd[1] := i*0.02;
            if j = 1 then tabq[i, 0] := _wd[1]; { Zapis kolejnych wartości współczynnika kr }
            tabq[i, j] := wskaznik(_wd); { Zapisywanie wyników w pomocniczej tablicy rozwiązań }
        end;
    end;
    { Zamiana tablicy na plik rozwiazań przy użyciu pomocniczej procedury z modułu DarPDod }
    pokdod ( @tabq, nkr, ntr+1, 'kr q0 q20 q40 q60 q80 q100', 'tabq', 1, 1);
end;
{$F-}
begin   {******* Opis badań symulacyjnych ******}
    Diffeq := OpisUkladu;           QFunc := wskaznik;          RunAction1 := Akcja1;
    SetT0Tmax(0.0, 1000.0);     { Ustawianie przedziału całkowania }
    SetHHout(0.05, 0.0);        { Ustawianie początkowego kroku całkowania }
    interact; { Uaktywnienie trybu pracy interakcyjnej }
end.
```

W trybie pracy interakcyjnej dokonywane jest przeglądanie modelu, ustawianie parametrów eksperymentu oraz parametrów modelu, inicjowanie wykonania sekwencji eksperymentów (wywołanie procedury RunAction1) i obserwacja uzyskanych wyników.

Korzystając z modelu sekwencyjnego można łatwo utworzyć model procesu *Nadawcy* i model procesu *Wykonawcy*. W modelu *Nadawcy* inaczej należy przygotować procedurę Akcja1. W procedurze tej należy jawnie rozdzielić fazy: przygotowania danych sekwencji

eksperymentów, wykonania sekwencji eksperymentów i kompletowania wyników. Przykładowy opis modelu *Nadawcy* przedstawia następujący wydruk:

```
program model1_nad; { Model procesu Nadawcy }
{$M 16384,10000,40000}
uses Dos, Crt, DarVar, DarRun, DarKom, DarPDod, DarOpt;
Const
   nkr = 16;   ntr = 6; { Pomocnicze stałe określające rozmiar tablicy rozwiązań }
Var
   Xuser:    record w,    y,    q,    x,    qq:   otyp;  end absolute _X;
   DXuser:   record dw,   dy,   dq,   dx,   dqq:  otyp;  end absolute _DX;
   YUser:    record s,    e,    u:    otyp;  end absolute _Y;
   Puser:    record kr,   tr:   otyp;  end absolute _P;
   t:        otyp absolute _T; { zmienna niezależna }
{$F+}
Procedure OpisUkladu; { Zapis równań stanu }
begin
   with XUser, DXUser, YUser, PUser do begin
      s := 1;              e := - q + s;              u := (tr*e + x)/0.05;
      dx := kr*e - u;      dw := (0.2*u - w)/ 50.0;   dy := w;
      dq := ( y - q )/40.0;  dqq := sqr(e);
   end;
end;
Function wskaznik(aa:_TabValueA): otyp; { Funkcja wyznaczająca wartość wskaźnika jakości }
begin
   with XUser, YUser, PUser do begin
      kr := aa[1];         tr := aa[2];      { Ustawianie parametrów modelu }
      Run;        { Wykonanie pojedynczego eksperymentu }
      wskaznik := qq;      { Wartość wskaźnika jakości }
   end;
end;
Procedure Akcja1; {     Procedura wyznaczania wartości tablicy danych sekwencji eksperymentów,
                        wywołania procedury równoległej realizacji sekwencji eksperymentów oraz
                        kompletowania wyników }
var
   tabq:       array[1..nkr, 0..ntr] of otyp; { Pomocnicza tablica rozwiązań }
   i, j, jj:   integer; { Pomocnicze zmienne sterujące }
begin
   jj := 1;  { Zapisanie danych do tablicy _ptd^ }
   for j := 1 to ntr do begin
      _wd[2] := (j-1)*20.0;
      for i := 1 to nkr do begin
         _wd[1] := i*0.02;        if j = 1 then tabq[i, 0] := _wd[1];
         _ptd^[jj][1] := _wd[1];  _ptd^[jj][2] := _wd[2];          inc(jj);
      end;
   end;
   Make(1, nkr*ntr, _MWyk); { Zlecenie wykonania sekwencji eksperymentów }
   jj := 1;      { Przepisanie wyników z tablicy _ptw^ do pomocniczej tablicy rozwiązań }
   for j := 1 to ntr do begin
      for i := 1 to nkr do begin   tabq[i,j] := _ptw^[jj].wynik[1];        inc(jj);  end;
   end;
   { Zamiana tablicy na plik rozwiazań przy użyciu pomocniczej procedury z modułu DarPDod }
   pokdod ( @tabq, nkr, ntr+1, 'kr g0 q20 q40 q60 q80 q100', 'tabq', 1, 1);
end;
{$F-}
```

```
begin    {******* Opis badań symulacyjnych ******}
    Diffeq := OpisUkladu;        QFunc := wskaznik;        RunAction1 := Akcja1;
    SetT0Tmax(0.0, 1000.0);   { Ustawianie przedziału całkowania }
    SetHHout(0.05, 0.0);      { Ustawianie początkowego kroku całkowania }
    SetNA(2);        { Liczba danych pojedynczego eksperymentu }
    Interact;  { Uaktywnienie trybu pracy interakcyjnej }
end.
```

Zmienna systemowa _MWyk, oznaczająca liczbę wykonawców, ustawiana jest w trybie pracy interakcyjnej.

W modelu *Wykonawcy* niepotrzebna jest procedura Akcja1, natomiast w części głównej programu zamiast wywołania procedury Interact umieszcza się wywołanie procedury Execution. W modelu *Wykonawcy* należy również zamiast procedury funkcyjnej wskaznik użyć procedury wskaznik2. Przykładowy opis model *Wykonawcy* przedstawia kolejny wydruk:

```
program model1_wyk; { Model procesu Wykonawcy }
{$M 16384,10000,40000}
uses Dos, Crt, DarVar, DarRun, DarKom, DarPDod, DarOpt;
Var
    Xuser:     record  w,     y,     q,     x,     qq:    otyp;   end absolute _X;
    DXuser:    record  dw,    dy,    dq,    dx,    dqq:   otyp;   end absolute _DX;
    YUser:     record  s,     e,     u:     otyp;  end absolute _Y;
    Puser:     record  kr,    tr:    otyp;  end absolute _P;
    t:         otyp absolute _T; { zmienna niezależna }
{$F+}
Procedure OpisUkladu; { Zapis równań stanu }
begin
    with  XUser, DXUser, YUser, PUser  do  begin
        s := 1;                  e := - q + s;                  u := (tr*e + x)/0.05;
        dx :=  kr*e - u;         dw := (0.2*u - w)/ 50.0;       dy := w;
        dq := ( y - q )/40.0;    dqq := sqr(e);
    end;
end;
Procedure wskaznik2(aa:_TabValueA; Var bb: _TabValueA); { Wyznaczanie wskaźnika jakości }
begin
    with XUser, YUser, PUser do begin
        kr := aa[1];             tr := aa[2];{ Ustawianie parametrów modelu }
        Run;          { Wykonanie pojedynczego eksperymentu }
        bb[1] := qq;      { Wartość wskaźnika jakości }
    end;
end;
{$F-}
begin    {******* Opis badań symulacyjnych ******}
    Diffeq := OpisUkladu;
    RunFunc := wskaznik2;
    SetT0Tmax(0.0, 1000.0);   { Ustawianie przedziału całkowania }
    SetHHout(0.05, 0.0);      { Ustawianie początkowego kroku całkowania }
     Execution(1);   { Oczekiwanie na komunikat od Wykonawcy, wykonanie eksperymentu }
end.
```

5. Wyniki badań

W celu zaprezentowania efektywności równoległej realizacji sekwencji eksperymentów wykonano odpowiednie badania. Do badań użyto czterech stacji z procesorem 486DX2-66 oraz jednej stacji z procesorem 486DX-50. Jako charakterystykę dla porównania poszczególnych stacji przyjęto średni czas szeregowego wykonania sekwencji eksperymentów, uzyskany dla modelu pierwszego (tabela 1).

Tabela 1

Średni czas szeregowego wykonania sekwencji eksperymentów

Oznaczenie stacji	Procesor	Czas realizacji [s]
1	486DX2-66	208.8
2	486DX2-66	216.4
3	486DX2-66	227.5
4	486DX2-66	231.6
5	486DX-50	304.0

Podczas równoległej realizacji sekwencji eksperymentów proces *Nadawcy* uruchomiony był w stacji 5, natomiast proces *j*-tego *Wykonawcy* uruchomiony był w stacji *j* (*j*=1, 2, 3, 4). Uzyskane średnie czasy równoległego wykonania sekwencji eksperymentów oraz uzyskane przyśpieszenia przedstawiono w tabeli 2.

Tabela 2

Średni czas i przyśpieszenie równoległego wykonania sekwencji eksperymentów

Liczba wykonawców i	1	2	3	4
Czas realizacji w [s] t_{ri}	214.3	109.0	75.4	57.1
Przyśpieszenie p_i	0.97	1.95	2.89	3.87

Przyśpieszenie określono według wzoru:

$$p_i = \frac{\sum_{j=1}^{i} t_j}{i \cdot t_{ri}} \tag{3}$$

gdzie:

p_i – przyśpieszenie dla *i Wykonawców*,

t_j – czas szeregowej realizacji sekwencji eksperymentów przez *j*-tego *Wykonawcę*,

t_{ri} – czas równoległej realizacji sekwencji eksperymentów przez *i Wykonawców*.

Porównując wyniki przedstawione w tabelach 1 i 2 można również określić czas niezbędny na transmisję danych między procesem *Nadawcy* a procesami *Wykonawców*. Czas ten jest największy, gdy badanie wykonane jest w układzie *Nadawca* i jeden *Wykonawca*. W tym przypadku wynosi on 5.5 [s] na realizację sekwencji eksperymentów lub średnio 0.057 [s] na eksperyment, przy średnim czasie realizacji eksperymentu równym 2.18 [s]. W rozpatrywanym przypadku czas wykonania eksperymentu jest znacznie większy od czasu transmisji i w takich przypadkach uzyskuje się największe przyśpieszenia, prawie równe liczbie *Wykonawców*.

6. Uwagi końcowe

W pracy przedstawiono tylko te elementy pakietu DarP, które ilustrują sposób tworzenia modeli procesu *Nadawcy* i procesu *Wykonawcy*. Tworząc takie modele użytkownik programowo ustawia te parametry eksperymentu i te parametry modelu, które są stałe podczas realizacji sekwencji eksperymentów oraz jawnie określa które i jakie jest znaczenie parametrów przekazywanych między procesem *Nadawcy* i procesami *Wykonawców*. Uwzględniając koszt (czas) transmisji danych uzasadnione jest tu takie tworzenie modeli, w których wielkość przesyłanych danych jest najmniejsza. Przyjęcie takiego założenia powoduje, że zakres stosowalności modelu *Wykonawcy* jest ograniczony do ustawionych programowo parametrów.

W celu pozbycia się wspomnianego ograniczenia oraz umożliwienia w pełni interakcyjnej realizacji badań modelowych w systemie wielostanowiskowym wprowadzono w pakiecie DarP dodatkowe mechanizmy, umożliwiające:

- przesyłanie wartości parametrów ustawionych w modelu *Nadawcy* do modeli *Wykonawców*,
- zapisywanie plików raportów utworzonych w procesach *Wykonawców* do wskazanych plików wyjściowych,
- usuwanie (kończenie) procesów *Wykonawców*.

Przed rozpoczęciem badań modelowych, z równoległym wykonywaniem eksperymentów na poszczególnych stacjach sieci komputerowej, użytkownik musi „ręcznie" zainicjować procesy *Wykonawców*, które od tego momentu są w stanie oczekiwania na komunikat od *Nadawcy*. Ostatni z wymienionych dodatkowych mechanizmów umożliwia automatyczne zakończenie przebiegu procesów *Wykonawców*.

Pakiet DarP umożliwia również zapisywanie wyników eksperymentów do tzw. plików rozwiązań [5]. Taki efekt działania procesu *Wykonawcy* uzyskuje się ustawiając odpowiednie flagi systemowe. Nie bez znaczenia jest tu również cecha pakietu DarP, polegająca na automa-

tycznej generacji unikalnych nazw dla kolejnych plików rozwiązań. Informacje o nazwach plików rozwiązań mogą być zapisywane w plikach raportów lub odtwarzane na podstawie zawartości pola *nrwyk* rekordu tablicy wyników, która wskazuje numer katalogu *Wykonawcy*.

LITERATURA

[1] Wilk A.: Koncepcja wykorzystania sieci „NetWare" do realizacji przetwarzania równoległego. ZN Pol. Śl. s. Informatyka, z. 26, Gliwice 1994.

[2] Węgrzyn S.: Systemy sterowane przepływem operacji i systemy sterowane przepływem argumentów. ZN Pol. Śl. s. Informatyka, z. 24, Gliwice 1993.

[3] Skowronek M.: Wykorzystanie sieci NetWare do równoległej realizacji zadań modelowania. ZN Pol. Śl. s. Informatyka, z. 27, Gliwice 1994.

[4] Skowronek M.: Równoległa realizacja algorytmów rozwiązywania zadań optymalizacji parametrycznej. Archiwum Informatyki Teoretycznej i Stosowanej, w druku.

[5] Skowronek M.: Modelowanie cyfrowe układów dynamicznych. Wyd. II, Skrypt Pol. Śl. nr 1776, Gliwice 1993.

Recenzent: Dr hab. inż. Stanisław Wołek

Wpłynęło do Redakcji 15 stycznia 1996 r.

Abstract

The modelling tasks require a lot of time because a great number of experiments are necessary. One of the ways to limit this time is parallel execution of the experiments in the separate stations of a computer network. In the paper were presented the rules of coordination as well as some as examples of synchronization and communication algorithms for the parallel cooperating processes. On the base of presented algorithms analysis were defined the parameters and initial data of the following procedures:

- the execution of the sequence of the experiments procedure (*Sender* model),
- the waiting for massage and experiment execution procedure (*Executor* model).

The description of new items of the DarP package useful for building parallel experiments model and some examples of their usage were also included in the paper.

To estimate execution performance of the sequence of the experiments some models were created and the time of the sequence of the experiments was measured for serial and parallel

execution. In the table 1 was presented the execution time of sequence of experiments for the separate station. In the table 2 was presented the parallel execution time as a function of the *Executors number* and the respective speed up (equation 3).

The modifications of the DarP package allows also interactive execution of the modelling tasks in multi station system.