RAČUNARSTVO
Sveska 1, Broj 2, 1992

UDK 519.683

PARALELIZACIJA RADA GENERATORA PSEUDO-SLUČAJNIH NIZOVA BITOVA (GPSN) I UOPŠTENOG GENERATORA PSEUDO-SLUČAJNIH BROJEVA (CFSR)

TANJA PULEVIĆ

SAŽETAK. U ovom članku se razmatra paralelizacija rada nekih generatora slučajnih brojeva. Tehnički, akcenat je stavljen na simulaciju dijeljenja memorijskog prostora kod transpjuterske ploče sa procesorima T800.

Prije izlaganja algoritama, nekih teorijskih osnova i razloga zbog kojih su generatori pseudo-slučajnih brojeva čiji rad paralelizujemo interesantni, upoznajmo se sa paketom potprograma Paralelnog Fortrana koji omogućava simulaciju dijeljenja memorijskog prostora.

1 O nitima u paralelnom Fortranu

Transpjuterska ploča sa procesorima T800 na kojoj su implementirani algoritmi iz ovog rada, nema realnu mogućnost dijeljenja memorijskog prostora među procesima. Paralelni Fortran koristi prednosti arhitekture transpjutera i dozvoljava kreiranje niti koje se konkurentno izvršavaju unutar jednog zadatka. Niti jednog zadatka izvršavaju se na jednom procesoru i mogu dijeliti memorijski prostor. Osim preko zajednicke memorije, niti mogu komunicirati jedna sa drugom i preko kanala. Potprogrami paketa `THREAD` (nit) omogućavaju programu napisanom na Paralelnom Fortranu da kreira niti unutar jednog zadatka. Svaki potprogram koji koristi paket `THREAD` treba da uvede datoteku paketa, koja se zove `THREAD.INC`, pomoću rečenice

```
INCLUDE 'THREAD.INC'
```

Opišimo neke potprograme i funkcije paketa `THREAD`, tj. ono što nam je potrebno za elementarne manipulacije sa nitima.

Kreiranje niti

Nit se može kreirati pomoću opšteg potprograma F77_THREAD_START ili pomoću funkcije F77_THREAD_CREATE.

Poziv

```
CALL F77_THREAD_START(SUB,WSARRAY,WSSIZE,FLAGS,NARGS,ARG1, ... ,ARGN)
```

znači da se kreira i počinje da se izvršava nova nit koja je bazirana na opštem potprogramu SUB. Ova nit koristi radni prostor koji počinje nizom WSARRAY. WSARRAY je tipa INTEGER. Veličina radnog prostora u bajtima je opisana sa WSSIZE. Argument FLAGS odgovara skupu atributa niti koja se kreira. Za sada, na raspolaganju je samo jedan atribut, proritet niti. Prioritet može biti "urgentan" i "nije urgentan". U datoteci THREAD.INC date su konstante koje opisuju atribut prioriteta. Te konstante su: F77_THREAD_URGENT i F77_THREAD_NOTURG. Ako nova nit treba da ima isti prioritet kao tekuća (ona u čijem sklopu se kreira nova nit) za argument FLAGS se uzima vrijednost F77_THREAD_PRIORITY() koja je tipa INTEGER. NARGS odgovara broju argumenata koje predajemo potprogramu SUB. ARG1, ... ,ARGN su argumenti koji se predaju potprogramu. Argumenti mogu biti bilo kojeg tipa samo treba obratiti pažnju da se promjenljive tipa CHARACTER predaju potprogramu pomoću dva argumenta i o tome treba voditi racuna kada se specificira NARG.

Drugi način da se kreira nit je pomoću logičke funkcije F77_THREAD_CREATE. Poziv I=F77_THREAD_CREATE(SUB,WSSIZE,NARGS,ARG1, ... ,ARGN) kreira i startuje potprogram SUB kao novu nit koja se izvršava sa istim prioritetom kao nit koja je kreira i radni prostor joj je veličine WSSIZE. Ostali argumenti imaju isto značenje kao kod F77_THREAD_START. Ako je nit kreirana, ova funkcija vraća vrijednost .TRUE., inače vraća vrijednost .FALSE..

Zaustavljanje niti

Nit završava svoje izvršavanje kada se SUB kao potprogram završi ili pomoću poziva CALL F77_THREAD_STOP.

Korišćenje RTL

Poziv CALL F77_THREAD_USE_RTL rezervise RTL (Run time library) za tekuću nit. Poziv CALL F77_THREAD_FREE_RTL oslobađa RTL.

Ovo bi bili osnovni pozivi.

2 O generatorima pseudo-slučajnih nizova bitova

U kriptografiji se sa posebnim interesovanjem izučavaju pseudo-slučajni nizovi bitova, kao i specijalni hardver namijenjen za njihovo generisanje. Generatori pseudo-slučajnih nizova bitova (u oznaci GPSN) su bazirani na pomjeračkim registrima. Oni proizvode i-ti bit u nizu na osnovu prethodnih m bitova na sljedeći način.

$$b_i = F(b_{i-1}, \ldots, b_{i-m})$$

za neku funkciju $F: \{0,1\}^m \mapsto \{0,1\}$

Obično se uzima linearna rekurentna veza

(1) $$b_i = (d_1 b_{i-1} + \cdots + d_k b_{i-m}) \mod 2$$

gdje su $d_1, \ldots, d_k$ binarne konstante. Sabiranje po modulu 2 i EOR (ekskluzivno ili) imaju istu istinitosnu tablicu, pa (1) može da se zapise kao

$$b_i = b_{i-l_1} \text{EOR} \, b_{i-l_2} \ldots b_{i-l_k}$$

gdje je $d_{l_1} = \cdots = d_{l_k} = 1$ i ostali $d_l = 0$. Uvedimo oznaku $\dot{+}$ umjesto EOR. Maksimalna perioda ovakvog generatora je $2^m - 1$. U principu, zanimljivi su oni generatori koji imaju maksimalnu periodu. Računanje periode kod generatora tipa (1) i razmatranje pitanja dostizanja maksimalne periode, obavlja se pomoću metoda faktorizacije polinoma nad konačnim poljima.

Rekurentnoj vezi (1) pridružuje se polinom

$$P(x) = x^m + d_1 x^{m-1} + \cdots + d_m$$

Uobičajeno je da se razmatraju trinomi

$$1 + x^r + x^s \quad \text{gdje je} \quad 1 \leq r < s$$

Ovom trinomu odgovara rekurentna veza

(2) $$b_i = b_{i-s} \dot{+} b_{i-(s-r)}$$

Obrtanjem niza dobija se trinom

$$1 + x^{s-r} + x^s$$

čija odgovarajuca rekurentna veza izgleda

(3) $$b_i = b_{i-s} \dot{+} b_{i-r}$$

(2) i (3) imaju istu periodu.

Ovdje nećemo razmatrati pitanje biranja parametara r, s i početnih bitova, jer nas interesuje paralelizacija rada generatora onda kada su svi parametri izabrani.

Navedimo neke parove (s, r) za koje GPSN daje maksimalnu periodu $2^s - 1$.

s	r
2	1
3	1, 2
⋮	⋮
17	3, 5, 6, 11, 12, 14
⋮	⋮
36	11, 25

Primijetimo da se od pseudo-slučajnih nizova bitova malim modifikacijama mogu dobiti pseudo slučajni brojevi iz uniformne raspodjele na intervalu (0,1). Postoji više načina za to. Jedan od najjednostavnijih je

$$U_i = 0.b_{i_t} b_{i_t+1} \ldots b_{i_t+n}$$

gdje su n i t prirodni brojevi takvi da je $0 < n < t$.

Generisanje brojeva iz uniformne raspodjele je, izmedu ostalog, interesantno jer se pomoću njih mogu modelirati druge raspodjele.

2.1 Paralelizacija rada GSPN

Cilj je, dakle, da se konstruiše paralelni algoritam koji proizvodi bitove pomoću rekurentne veze

$$b_i = b_{i-s} \dot{+} b_{i-r}$$

gdje su dati s, r i prvih s bitova. Izložicemo jednu ideju za $s = 9$, $r = 5$ i nproc $= 4$ (nproc je broj procesora). Ideja se uz male izmjene može primijeniti za neke druge s, r i nproc.

Paralelnu verziju rada GSPN

$$b_i = b_{i-9} \dot{+} b_{i-5} \tag{4}$$

opišimo algoritmom SHIFT1. Fortranski kod za ovaj algoritam dat je u prilogu. Ideja za SHIFT1 se prirodno nameće rekurentnom vezom (4) i zahtijeva dijeljenje memorije među procesorima. To će se simulirati. Algoritam je jednostavan, njegovi zahtjevi za dijeljenje memorije se zadovoljavaju lako i zato je dobar kao edukativan primjer za početak rada sa paketom THREAD koji smo opisali u dijelu 1.

```
!
!Konfiguraciona datoteka za zadatak shift1
!
!  shift1.cfg configuration file for task shift1
!
!  Hardware
!
processor host
processor t1
processor t2
processor t3
wire ?  host[0] t1[0]     !anonymous wire connecting PC to transputer
wire ?  t1[1] t2[0]
wire ?  t1[2] t3[0]
!
!  Task declarations indicating channel I/O ports
!       and memory requirements
!
task afserver ins=1 outs=1
task filter ins=2 outs=2 data=10K
task shift1 ins=5 outs=5 data=500k
!
!  Assign software tasks to physical processors
!
place afserver host
place shift1 t1
```

```
    place filter t1
    !
    !  Set up the connections between the tasks.
    !
    connect ?  afserver[0] filter[0]
    connect ?  filter[0] afserver[0]
    connect ?  filter[1] shift1[1]
    connect ?  shift1[1] filter[1]
```

```
        PROGRAM SHIFT1
*************************************************************************
c       Vrsi se paralelizacija rada generatora slucajnih bitova.Simulira se
c       rad cetri procesa P1,P2,P3 i P4 koji dijele zajednicki prostor kroz
c       common zonu ZONA1.
c       Glavna nit P1 kreira niti P2,P3 i P4 koje imaju isti prioritet kao
c       ona.
c       niz bitova koji se generisu je niz DATA
*************************************************************************
c       Uvodjenje datoteke paketa THREAD
        INCLUDE 'thread.inc'
c       Radni prostori niz2,niz3 i niz4 za niti P2,P3,P4
        INTEGER niz2(2500),niz3(2500),niz4(2500)
        INTEGER data
        EXTERNAL P2,P3,P4
c       Definisanje zajednicke zone
        COMMON/zona1/data(300),ip2,ip3,ip4
c           Prioritet tekuce niti koji kao argument treba predati ostalima
        iprio=f77_thread_priority()
        PRINT *,'duzina niza ?'
        READ *,iduz
c       postavljanje pocetnih bitova
        DO 130 i=1,9
130             data(i)=1

c       Kreiranje niti P2,P3 i P4
        CALL F77_THREAD_START(P2,niz2,2500*4,iprio,1,iduz)
        CALL F77_THREAD_START(P3,niz3,2500*4,iprio,1,iduz)
        CALL F77_THREAD_START(P4,niz4,2500*4,iprio,1,iduz)
        PRINT *,'kreirao sam'
c       Proces (nit) P1 radi svoj dio posla
        DO 122 i=10,iduz,4
        data(i)=mod(data(i-9)+data(i-5),2)
122     CONTINUE
c       Da li su ostali zavrsili posao?  ako jeste stampanje niza
```

```
900         IF((ip2+ip3+ip4).eq.3)THEN
                PRINT *,(data(i),i=1,iduz)
                STOP
            ELSE
                GO TO 900
            ENDIF
        END

c***************PROCES P2*****************************
        SUBROUTINE P2(KK)
        INTEGER data
        COMMON/zona1/data(300),ip2,ip3,ip4
        ip2=0
        DO 100 i=11,kk,4
           data(i)=mod(data(i-9)+data(i-5),2)
100     CONTINUE
        ip2=1
        END

c***************PROCES P3*****************************
        SUBROUTINE P3(kk)
        INTEGER data
        COMMON/zona1/data(300),ip2,ip3,ip4
        ip3=0
        DO 100 i=12,kk,4
           data(i)=mod(data(i-9)+data(i-5),2)
100     CONTINUE
        ip3=1
        END

c***************PROCES P4*****************************
        SUBROUTINE P4(kk)
        INTEGER data
        COMMON/zona1/data(300),ip2,ip3,ip4
        ip4=0
        DO 100 I=13,KK,4
           data(i)=mod(data(i-9)+data(i-5),2)
100     CONTINUE
        ip4=1
        END
```

Ideja:

Počevši sa 9 datih bitova $b_1, \ldots, b_9$ četiri procesa generišu po jedan bit i tako, u jednom taktu, nastavljaju niz za četiri bita. Npr. u prvom taktu proces $P1$ generiše b_{10}, $P2$ generiše b_{11}, $P3$ generiše b_{12}, $P4$ generiše b_{13}.

Vrijednosti koje su date za s, r i nproc su takve da je ovo izvodljivo, jer u jednom taktu svaki proces ima različito polje rada.

Primijetimo da poslije nekoliko taktova kada se "istroše" početne vrijednosti, za rad procesa $P1$ potrebni su samo bitovi koje generiše proces $P4$. Analogno, proces $P2$ zavisi od procesa $P1$, proces $P3$ zavisi od procesa $P2$ i proces $P4$ zavisi od procesa $P3$.

Pretpostavili smo idealnu situaciju, da svaki proces obavlja svaku operaciju u istom vremenskom intervalu. Bez ove pretpostavke algoritam SHIFT1 treba da se dopuni uvođenjem "čekanja" među procesima. Procesi bi trebalo da se čekaju po principu zavisnosti koju smo naveli, tj. proces $P1$ bi čekao proces $P4$, proces $P2$ bi čekao proces $P1$ itd ... Čekanje se može realizovati semaforskom tehnikom, indikatorima u zajedničkoj memoriji i slično.

Iz izloženog se nameće da je ocjena efikasnosti algoritma optimalna tj. jednaka $O(\text{nproc})$.

3 O generalizovanom generatoru pseudo-slučajnih brojeva baziranom na pomjeračkom registru (oznaka GFSR)

Konstruktori GFSRA-a, Lewis i Payne, ostaju u svijetu bitova i princip rada ovog generatora zasnivaju na pravljenju n-bitnih prirodnih brojeva. Preciznije iz pseudo-slučajnog niza bitova uzima se n nesusjednih bitova i dobija se n-bitni prirodni broj

$$X_i = b_i b_{i-j_2} \dots b_{i-j_n} \tag{5}$$

Svaki bit od X_i zadovoljava zakon (2) pa se može formirati rekurentna veza $X_i = X_{i-s} + X_{i-(s-r)}$ Prvih s početnih vrijednosti ne moraju da zadovoljavaju vezu (5). Perioda niza niza (X_i) zavisi od početnih vrijednosti.

Trivijalan način da se od ovakvog niza dobije niz pseudo-slučajnih brojeva iz uniformne raspodlele na intervalu (0,1) jeste

$$U_i = 2^{-n} X_i$$

Primjer za GFSR $s = 7$, $r = 1$, $n = 3$ i početna matrica (seed) je:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix}$$

Ovaj generator daje niz cija je perioda jednaka 127:

0 1 2 3 4 5 6 1 3 1 7 1 3 7 2 2 6 6 2 4 ...

Algoritam koji je paralelna verzija rada GFSR-a i čiji fortranski kod je dat u prilogu, naziva se SHIFT2. Algoritam je ilustrovan na gornjem primjeru.

```
! konfiguraciona datoteka za shift2
! shft2.cfg configuration file for task shift2
!
! Hardware
!
processor host
processor t1
processor t2
processor t3
wire ? host[0] t1[0]    !anonymous wire connecting PC to transputer
wire ? t1[1] t2[0]
wire ? t1[2] t3[0]
!
! Task declarations indicating channel I/O ports
!     and memory requirements
!
task afserver ins=1 outs=1
task filter ins=2 outs=2 data=10K
task shi21 ins=5 outs=5 data=500k
!
! Assign software tasks to physical processors
!
place afserver host
place shi21 t1
place filter t1
!
! Set up connections between the tasks.
!
connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] shi21[1]
connect ? shi21[1] filter[1]
```

```
      PROGRAM SHIFT2
************************************************************************
C     Vrsi paralelizaciju rada uopstenog generatora slucajnih brojeva, pomocu
c     3 - bitnih prirodnih brojeva.  Simulira se rad tri procesa P1, P2 i P3.
c     Proces (nit) P1 kreira P2 i P3 koji su istog prioriteta kao i ona.
c     Matrica mat sadrzi 3-bitne prirodne brojeve
c     P1 formira prirodne brojeve od vrsta matrice mat koja je u djeljivoj
```

```
c       memoriji koju smo simulirali preko common zone zona1
c       Napomena :  pomjeranja se mogu parametrizovati.  Dakle, vrijednosti
c       za s i r se mogu ucitavati i prenijeti nitima kao ulazni argumenti.
c       Ovdje to nije uradjeno nego program radi za s=7 i r=1

***************PROCES P1 ************************************************
c       Proces P1 kreira ostale, radi na prvoj koloni, formira prirodne brojeve
c       i stampa.
C       Uvodjenje datoteke paketa THREAD
        INCLUDE 'thread.inc'
c       Radni prostori za P2 i P3
        INTEGER niz2(2500),niz3(2500)
        EXTERNAL P2,P3
        DIMENSION niiz(3000)
c       Definisanje djeljive memorije
            COMMON/zona1/mat(300,3),ipp2,ipp3,ipp4
c       Pocetne vrste matrice
        mat(1,1)=0
        mat(1,2)=0
        mat(1,3)=0
        mat(2,1)=0
        mat(2,2)=0
        mat(2,3)=1
        mat(3,1)=0
        mat(3,2)=1
        mat(3,3)=0
        mat(4,1)=0
        mat(4,2)=1
        mat(4,3)=1
        mat(5,1)=1
        mat(5,2)=0
        mat(5,3)=0
        mat(6,1)=1
        mat(6,2)=0
        mat(6,3)=1
        mat(7,1)=1
        mat(7,2)=1
        mat(7,3)=0

        PRINT *,'koliko vrsta ?'
        READ *,ivrs
c       Definisanje prioriteta tekuce niti
        IPRIO=F77_THREAD_PRIORITY()
        PRINT *,((mat(I,J),J=1,3),I=1,7)
c       Kreiranje niti P2 i P3
```

```
      CALL F77_THREAD_START(P2,NIZ2,2500*4,IPRIO,1,IVRS)
      CALL F77_THREAD_START(P3,NIZ3,2500*4,IPRIO,1,IVRS)

      PRINT *,'Kreirao sam' _
c     P1 obavlja svoj dio posla, radi na prvoj koloni
      DO 111 i=8,ivrs
111       mat(i,1)=mod(mat(i-7,1)+mat(i-6,1),2)
c     Da li su P2 i P3 zavrsili posao?  Ako jesu onda formiram prirodne brojeve
c     i stampam ih (zbog provjere)
9000      IF(ipp2+ipp3.eq.2)THEN
                DO 112 i=1,ivrs
                   niiz(i)=mat(i,1)*4+mat(i,2)*2+mat(i,3)
                   PRINT *,niiz(i)
112             CONTINUE
              ELSE
                 GO TO 9000
          ENDIF
      END

*******************PROCES P2 RADI DRUGU KOLONU************************

      SUBROUTINE P2(KK)
      COMMON/zona1/mat(300,3),ipp2,ipp3
      ipp2=0
      DO 22 i=8,kk
         mat(i,2)=mod(mat(i-7,2)+mat(i-6,2),2)
22          CONTINUE
      ipp2=1
      END

*******************PROCES P3 RADI TRECU KOLONU*******************************

      SUBROUTINE P3(KK)
      COMMON/zona1/mat(300,3),ipp2,ipp3
      ipp3=0
      DO 22 i=8,kk
         mat(i,3)=mod(mat(i-7,3)+mat(i-6,3),2)
22          CONTINUE
      ipp3=1
      END
```

3.1 Paralelizacija rada generatora GFSR

Dakle, treba paralelizovati generisanje niza n-bitnih prirodnih brojeva po vezi $X_i = X_{i-s} + X_{i-(s-r)}$. Algoritam SHIFT2 podrazumijeva da je nproc $= n$.

Ideja:

Izračunavanja po modulu 2 se vrše po koordinatama. Prvi procesor sabira bitove prve kolone po modulu 2, ..., n-ti procesor sabira bitove n-te kolone po modulu 2. Prilikom generisanja bitova procesi nemaju potrebe da se čekaju jer jedan proces radi stalno na jednoj koloni, kojoj ne pristupaju ostali procesi. Iz izloženog proizilazi ocjena efikasnosti rada algoritma: optimalna je i jednaka $O(\text{nproc})$.

Pretpostavka o n procesa može se zamijeniti sa $\text{nproc} = k$ gdje je $k < n$. Tada bi svaki proces radio na jednom bloku kolona. Blokovi su veličine $[n/k]$, eventualno neki blok bi bio veći. Ideja ostaje ista.

Ovaj rad je nastao zahvaljujući uslovima koji su mi obezbijedili MATEMATIČKI INSTITUT—BEOGRAD i MATEMATIČKI FAKULTET—BEOGRAD.

LITERATURA

[1] Brian D. Ripley, *Stochastic Simulation*, J Wiley & Sons, Canada, 1987.

Prirodno–matematički fakultet
Cetinjski put b. b.
81000 Titograd