Programowanie współbieżne

Wykłady - Cz. 5ex. Przykład. Liczby pierwsze.

Andrzej Baran
baran@kft.umcs.lublin.pl

Liczby pierwsze I

```
!
!! Program: pierwsze.f90 - znajdowanie liczb pierwszych w przedziale 2..n
!
! Metoda: przesiewanie (Eratostenes);
!         dekompozycja danych wejsciowych;
!
! Uwaga.
! Do wyznaczenia liczb pierwszych z przedzialu B=[floor(sqrt(n))+1..n]
! wystarcza znajomosc liczb pierwszych z przedzialu A=[2..floor(sqrt(n))]
! Kazda liczba zlozona z B dzieli sie bowiem przez jedna lub wiecej liczb
! pierwszych z A. Liczby pierwsze z A bedziemy nazywac podzielnikami.
! Jesli liczba z A dzieli sie przez podzielnik to jest zlozona.
!
! Przyklad.
! Znalezc liczby pierwsze z przedzialu A=[2..10000]. Najpierw znajdujemy
! liczby pierwsze z przedzialu [2..100]. Dla dowolnego j@B (@=nalezy)
! nalezy sprawdzic czy j dzieli sie przez ktorys z podzielnikow, ktorymi
! sa w tym przypadku liczby 2, 3, 5, ..., 97 (razem 25 liczb).
!
! W pierwszym kroku znajdujemy podzielniki (w kazdym procesie; w ten
! sposob oszczedzamy na komunikacji). W kroku drugim dokonujemy dekompozycji
! danych: przedzial B dzielimy na p podprzedzialów, gdzie p jest liczba
! procesow, o dlugosci d=ceiling((n-floor(sqrt(N)))/p).
! Procesy 0, 1, 2, ..., p otrzymuja podprzedzialy [s+1..s+d], [s+1+d..s+2d],
! ...,[s+1+(p-1)d..n], gdzie s=floor(sqrt(n)). Liczby liczb pierwszych
! sa zbierane w procesie 0 w tablicy liczby_grom z pomoca MPI_Gather().
! Ponieważ liczby liczb pierwszych, gromadzone w procesie 0,
! w poszczególnych podprzedzialach sa rożne, wiec procedura MPI_Gather
! tego nie zrobi (bo wymaga ona by dane gromadzone mialy ten sam rozmiar)
```

Liczby pierwsze II

```
! i nalezy uzyc procedury MPI_Gatherv().
!
program liczby_pierwsze
  use mpi

  ! definiowanie przedzialu 2..n
  integer, parameter :: n=10000000
  integer, parameter :: s=sqrt(float(n))
  integer, parameter :: m=n/10

  integer, dimension (0:s) :: a ! tablica pomocnicza
  integer, dimension (0:s) :: podzielniki ! podzielniki z przedzialu 2..sqrt(n)
  integer, dimension (0:m-1) :: pierwsze ! liczby pierwsze w podprzedzialach
  integer :: liczba, reszta, dl_podprz  ! dlugosc podprzedzialu
  integer :: id, p    ! id procesu i liczba procesow
  integer :: llpier   ! liczba liczb pierwszych
  integer, dimension (:), allocatable :: liczby_grom ! tab. liczb gromadzonych
  integer, dimension (:), allocatable :: przem ! tablica przemieszczen

  double precision :: czas   ! MPI_Wtime() func. is double...

  call MPI_Init( ierr )
  call MPI_Comm_Rank(MPI_COMM_WORLD, id, ierr)
  if ( id == 0 ) czas = MPI_Wtime()
  call MPI_Comm_Size(MPI_COMM_WORLD, p, ierr)

  ! wyznaczanie podzielnikow z przedzialu 2..s
  do i=2,s
     a(i)=1 ! inicjowanie
  end do
  i=1
```

Liczby pierwsze III

```
llpier=0
lpodz=0
do while ( i < s )
   i=i+1
   if ( a(i) == 1 ) then
      pierwsze(lpodz)=i        ! zapamietanie liczby pierwszej
      podzielniki(llpier)=i    ! ,, podzielnika
      lpodz=lpodz+1
      llpier=llpier+1
   end if
   ! wykreslanie liczb zlozonych, wielokrotnosci i
   do k=i+i, s, i
      a(k)=0
   end do
end do
!
! rownolegle wyznaczanie liczb pierwszych w podprzedzialach
!
dl_podprz=(n-s)/p ! obliczanie dlugosci podprzedzialu
if ( mod(n-s, p) /= 0 ) dl_podprz=dl_podprz+1
if ( id > 0 ) llpier=0  ! zerowanie licznika l. pierwszych w procesach > 0
!
do liczba=s+1 + dl_podprz*id, s + dl_podprz*(id+1)
   if ( liczba <= n ) then
      do k=0, lpodz-1
         reszta=mod(liczba, podzielniki(k))
         if ( reszta == 0 ) exit  ! liczba zlozona
      end do
      if ( reszta /= 0 ) then
         pierwsze(llpier)=liczba ! zapamietanie liczby pierwszej
         llpier=llpier+1
```

Liczby pierwsze IV

```
        end if
     end if
  end do
  !
  allocate(liczby_grom(0:p-1), STAT=istatus)
  call MPI_Gather(llpier, 1, MPI_INTEGER, liczby_grom, 1, MPI_INTEGER, 0, &
       MPI_COMM_WORLD, ierr)
  allocate(przem(0:p-1), STAT=istatus) ! tablica przemieszczen
  przem(0)=0
  liczba_p=liczby_grom(0) ! calkowita liczba wyznaczonych liczb pierwszych
  do  i=1, p-1
     przem(i)=przem(i-1)+liczby_grom(i-1)
     liczba_p=liczba_p+liczby_grom(i)
  end do
  !
  ! gromadzenie liczb pierwszych w procesie 0;
  !    call MPI_Gatherv(pierwsze, llpier, MPI_INTEGER, pierwsze, liczby_grom, &
  !         przem, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
  write(*,*) id, llpier
  if ( id == 0 ) then
     czas = MPI_Wtime() - czas
     write(*,*) 'Czas =', czas, 'sec'
  end if

  deallocate(liczby_grom, przem, STAT=istatus)
  call MPI_FINALIZE(ierr)

end program liczby_pierwsze
```

Podstawa: Z. Czech: *Wprowadzenie do programowania równoległego. PWN, Warszawa, 2010.*

MPI_Gatherv(...)

```
 call MPI_GATHERV (
   wysylane_dane(:)                ! dane
   liczba_wysylanych_danych        ! INTEGER
   typ_wysylanych_danych           ! MPI_Datatype
   odbierane_dane( )               ! dane
   liczba_odbieranych_danych( )    ! INTEGER
   tablica_przemieszczen( )        ! INTEGER
   typ_odbieranych_danych          ! MPI_Datatype
   proces_glowny                   ! INTEGER
   komunikator                     ! MPI_COMM
)
```

Jak korzystać z tablicy przemieszczeń?

Dane z procesu 0 umieszczane są od pozycji `odbierane_dane+przemieszczenie(0)`, z procesu 1 od pozycji `odbierane_dane+przemieszczenie(1)` itd. Parametr `odbierane_dane` ma charakter wyjściowy (pozostale parametry to parametry wejściowe) i ma znaczenie tylko w procesie głównym (0).